

Institut für Informationswirtschaft
Wirtschaftsuniversität Wien UZA II 3.Ebene
Augasse 2-6, A-1090 Wien, Austria



Anbindung von Handhelds an SAP am Beispiel einer sprachgesteuerten Transaktion

Bakkalaureatsarbeit im Rahmen des IT-Praktikums
Bakkalaureat Wirtschaftsinformatik

Univ.Ass. Dr. Edward Bernroider
Betreuer

Bernhard Bugelmüller 9403085

WS 05/06

1 Einleitung

1.1 Zusammenfassung

Es gibt mehrere Möglichkeiten um Handhelds (PDAs oder Funkscanner) online an ein SAP System anzubinden. Die gebräuchlichsten Lösungen, die von SAP unterstützt werden, sind die *SAP Console*, die *SAP WEBConsole* und der *SAP Web Application Server*. Je nach Projektanforderung bieten die einzelnen Möglichkeiten konkrete Vor- und Nachteile.

Für sprachgesteuerte Transaktionen bieten sich vor allem die WEBConsole und der Web Application Server an. Die WEBConsole bietet den Vorteil, dass einige Standard-Transaktionen im Lieferumfang des SAP Systems bereits inkludiert sind. Weiters beschränkt sich das notwendige Know-how für Erweiterungsentwicklungen auf die Programmiersprache *ABAP* bzw. auf die Entwicklungsumgebung *ABAP Development Workbench*. Bei großem Umfang nötiger Erweiterungen sollte dem Web Application Server der Vorzug gegeben werden. Der anfängliche Programmieraufwand einer Transaktion ist zwar höher allerdings sind weitere Transaktionen schnell duplizierbar. Das benötigte Know-how stellen die Sprachen *HTML*, *JavaScript*, *VoiceXML* und ABAP sowie der Umgang mit *Business Server Pages* auf dem Web Application Server dar.

Grundsätzlich wird für sprachgesteuerte Transaktionen der *X+V Standard* verwendet. Dieser wird durch das *World Wide Web Consortium* (*W3C*) unterstützt und es gibt derzeit zwei kompatible Browser (*NetFront* und *Opera*). Diese sind zwar sehr preisgünstig, unterstützen allerdings nur Englisch und sind in erster Linie nicht für professionelle Anwendungen z.B. *Pick by Voice* mit lagertypischer Geräuschkulisse ausgerichtet. Für derartige Anwendungen empfehlen sich kommerzielle Anbieter von Spracherkennungssoftware.

1.2 Zielsetzung und Methodik

Wissenschaftliche Zielsetzung im Rahmen des Praktikums war es, einerseits festzustellen und zu erläutern, welche Kategorien von mobilen Endgeräten und welche Arten von drahtlosen lokalen Netzwerken existieren, andererseits zu analysieren, inwieweit Handhelds in der betrieblichen Informationsverarbeitung eingesetzt werden und welcher Nutzen sich daraus ergibt. Die praktische Zielsetzung bestand darin, herauszufinden welche technischen Möglichkeiten es gibt Handhelds an SAP Systeme anzubinden und eine sprachgesteuerte Beispieltransaktion zu entwickeln, die für das Entwicklerteam des Auftraggebers als Vorlage für zukünftige Projekte dienen soll. Um eine Beispieltransaktion selbständig entwickeln zu können, muss zuerst analysiert werden, welches Know-how in den Bereichen Sprachausgabe und Spracherkennung benötigt wird bzw. aufzubauen ist. Zur Entwicklung der Transaktion wurde ein, von der Flexus AG vorgegebenes und bereits in zahlreichen

Projekten angewandtes, Vorgehensmodell gewählt, welches in den Grundzügen mit dem klassischen Wasserfallmodell identisch ist und folgende Phasen aufweist:

i. Anforderungsdefinition
ii. Analyse (Grobdesign)
iii. Entwurf (Feindesign)
iv. Realisierung
v. Test
vi. Produktivsetzung

Als Entwicklungsumgebung diente der SAP Web Application Server (WAS) 6.4. Unter Verwendung der Sprachen HTML, ABAP, JavaScript und VoiceXML wurden einzelne Business Server Pages (BSPs) erstellt. Die Ablauflogik bzw. die Anbindung des Backend-Systems (R/3 WM) wurde mittels ABAP im Event-Handler des WAS programmiert.

1.3 Grundbegriffe und Definitionen

Die vorliegende Arbeit beschäftigt sich mit der Problematik der Online-Anbindung von Handhelds an SAP Systeme über Funk. Auf diesen Funkterminals sollen sprachgesteuerte Transaktionen realisiert werden. Die folgenden Grundbegriffe bzw. Definitionen dienen dem besseren Verständnis des Themengebiets.

SAP. Integrierte betriebswirtschaftliche Standardsoftware, basierend auf einzelnen Modulen (Finance, Controlling, Logistic Execution …), die sämtliche Bereiche des Unternehmens abdeckt.

TRANSAKTION. Ein logisch und betriebswirtschaftlich zusammenhängender Geschäftsablauf (z.B. Anlegen eines Kunden), der über ein ausführbares Programm mit einer oder mehreren Bildschirmmaske(n) realisiert wird.

TEXT TO SPEECH (TTS). Ein geschriebener Text wird in ein Sprachsignal umgewandelt. Grundsätzlich lassen sich zwei Ansätze zur Erzeugung eines Sprachsignals unterscheiden. Zum einen kann auf Sprachaufnahmen (*Samples*) zurückgegriffen werden, die passend verändert werden. Zum anderen kann das Signal aber auch vollständig im Rechner durch eine synthetische Stimme erzeugt werden (vgl. Wikipedia, 2005b).

VOICE RECOGNITION (VR). „Die Spracherkennung oder auch automatische Spracherkennung ist ein Teilgebiet der angewandten Informatik. Sie beschäftigt sich mit der Untersuchung und Entwicklung von Verfahren, die es Automaten, insbesondere Computern erlauben, gesprochene Sprache zu erkennen (das heißt, in Zeichenfolgen umzuwandeln) und zu verarbeiten.“ (Wikipedia, 2005c)

2 Theoretische Grundlagen

2.1 Mobile Endgeräte und drahtlose Netzwerke

2.1.1 Kategorien mobiler Endgeräte

Ziel mobiler Endgeräte ist es, mobilen Benutzern die Nutzung von Diensten über ein drahtloses Netzwerk oder lokal verfügbarer mobiler Anwendungen zu ermöglichen. Die Bandbreite derartiger Geräte reicht von kleinsten Spezialgeräten bis hin zu Computern mit der Leistungsfähigkeit von Desktop-PCs. Eine Klassifizierung dieser Geräte gestaltet sich sehr schwierig, da mobile Endgeräte in einem ernormen Tempo weiterentwickelt werden und ständig neue Typen bzw. Produkte auf den Markt kommen. Eine strenge Einteilung in einzelne Kategorien wird zusätzlich dadurch erschwert, dass der Trend in Richtung hybrider Geräte geht, die mehrere Funktionen in sich vereinen und daher auch zu mehreren Kategorien zugeordnet werden können. Roth (2002) lässt aufgrund der beschriebenen Problematik konkrete Leistungsmerkmale der Geräte außer Acht und beschränkt sich auf eine qualitative Beschreibung. Zur Klassifizierung mobiler Endgeräte werden fünf Kategorien eingeführt.

MOBILE STANDARDCOMPUTER. Diese verfügen nahezu über dieselbe Leistungsfähigkeit wie stationäre Computer. Es handelt sich um die mobile Ausführung von Standardcomputern.

BORDCOMPUTER. Es handelt sich um hochspezialisierte Computer die in Fahr- und Flugzeugen, Schiffen, Raumfahrzeugen und Satelliten fest installiert sind.

HANDHELDS. Kleine mobile Computer, welche gegenüber Standardcomputern über eine reduzierte Leistungsfähigkeit verfügen. Sie sind so klein, dass sie in einer Hand gehalten und im gehaltenen Zustand auch bedient werden können. Das ist auch der wesentliche Unterschied zu mobilen Standardcomputern, die in der Regel auf einer ebenen Unterlage aufgestellt werden müssen.

WEARABLES. Diese Endgeräte werden nicht in der Hand, sondern am Körper getragen. Dazu werden sie in die Kleidung integriert oder direkt am Körper, über ein Armband oder über eine Kopfbefestigung, getragen.

CHIPKARTEN. Die kleinste Variante mobiler Endgeräte. Chipkarten sind im Grunde genommen kein selbständiges Endgerät, da sie für den Einsatz ein Lesegerät benötigen. Sie verfügen aber über Speicher und Prozessor und können auch teilweise programmiert

werden. Ihr Einsatzgebiet liegt vorzugsweise im M-Commerce und in der Benutzeridentifikation.

Zusätzlich zu diesen fünf Kategorien ist eine weitere, davon unabhängige Einteilung in Universal- und Spezialgeräte sinnvoll.

Universalgeräte werden vom Hersteller nicht für einen bestimmten Zweck vorgesehen. Sie erlauben die Installation beliebiger Anwendungen. Notwendige Voraussetzung ist ein flexibles Betriebssystem und definierte Schnittstellen zum Laden neuer Programme und Daten. Entwickler dieser Programme müssen mit der entsprechenden Entwicklungsumgebung unterstützt werden.

Spezialgeräte werden für einen bestimmten Einsatzzweck hergestellt. Die Änderung der Programmierung ist nicht vorgesehen und meistens technisch unmöglich. Bei Spezilageräten werden bestimmte Hardware-Komponenten für den Zweck der Anwendung optimiert.

Werden die fünf Kategorien der Einteilung in Universal- und Spezialgeräte gegenübergestellt, erhält man Tabelle 1.

Tabelle 1: Klassifikation mobiler Endgeräte

<table>
<tr><th>Kategorie</th><th>Universalgerät</th><th>Spezialgerät</th></tr>
<tr><td>Mobile Standardcomputer</td><td>Notebook</td><td>Spezielle mobile Computer, z.B. in der Vermessungstechnik, Kartografie und Archäologie</td></tr>
<tr><td>Bordcomputer</td><td>-</td><td>Bordcomputer in Fahr- und Flugzeugen, Computer in Satelliten</td></tr>
<tr><td rowspan="2">Handhelds</td><td>PDA (Personal Digital Assistent)</td><td>Elektronischer Kalender (nicht programmierbar), Lesestift, E-Book, Web-Pad, mobiles Datenerfassungsterminal in der Lagerhaltung, GPS-Empfänger, Mobiltelefon, Pager, Digitalkamera</td></tr>
<tr><td colspan="2">Smart-Phone, Communicator, Mobile Spielekonsole, Programmierbarer Taschenrechner</td></tr>
<tr><td>Wearables</td><td>Programmierbares Wearable</td><td>Armbanduhr, Pulsmesser</td></tr>
<tr><td>Chipkarten</td><td>Smart Card</td><td>SIM-Karte, EC-Karte mit Bargeldfunktion, Telefonkarte, Identifikation zur Zeiterfassung, Karte für digitale Unterschrift</td></tr>
</table>

Quelle: Roth, 2002

Da im vorliegenden Praktikum ein Handheld von Hewlett Packard (HP iPAQ) zum Einsatz kommt soll an dieser Stelle noch ein wenig genauer auf diese Kategorie eingegangen werden.

Handhelds treten in sehr vielen Gerätevarianten auf. Wie bereits erwähnt ist allen Handhelds gemein, dass sie in der Hand gehalten werden können und keinen stationären Aufbau benötigen. Handhelds können somit auch benutzt werden, während man geht oder steht.

Die Universalgeräte aus dem Bereich Handhelds werden PDA (*Personal Digital Assistent*), Notepad oder Organizer genannt. In der Regel sind diese Geräte bereits mit vorinstallierten Anwendungen wie Termin- und Adressverwaltung sowie Notizbuchfunktion ausgestattet. Weitere Programme können vom Benutzer nach Belieben und Speicherkapazität installiert werden. In den meisten Fällen verfügt der PDA auch über eine Möglichkeit die Daten mit einem stationären Computer zu synchronisieren. Die Dateneingabe erfolgt entweder mittels (Mini-)Tastatur oder Stift und Handschrifterkennung. Tastatur- wie stiftbasierte PDAs haben folgende Eigenschaften:

- Sehr reduzierte Prozessorleistungen und geringere Speicherkapazität, verglichen mit Standardcomputern.
- In der Regel steht keine Festplatte zur Verfügung. Die Daten und Anwendungen sind deshalb im Festwertspeicher fest abgelegt oder werden in einem batteriegepufferten Speicher untergebracht.
- Die Anzeigen sind sehr klein und haben eine geringere Auflösung, verglichen mit Standardcomputern.
- In einen PDA ist der so genannte *Digitizer* integriert. Damit kann der Benutzer mit einem Stift die Benutzeroberfläche, ähnlich wie mit einer Maus, bedienen, indem Punkte der Anzeige angetippt werden.
- In die meisten Geräte können Erweiterungskarten eingesteckt werden. Damit kann zusätzlicher Speicherplatz genutzt oder der PDA um notwendige Funktionen erweitert werden. Es existiert eine Vielzahl von, oft zueinander inkompatiblen, Kartenformaten (z.B. *Compact Flash*, *Memory Stick*, *SpringBoard*, *Secure Digital*, *Smart Media* und *Multimedia Cards*).
- Die Stromversorgung von PDAs erfolgt auf Basis von Batterien oder Akkus.
- Als Kommunikationsschnittstellen stehen meist Infrarot- oder Bluetooth-Transceiver sowie serielle Anschlüsse zur Verfügung.

Aus Benutzersicht, werden PDAs auf andere Weise bedient wie z.B. Notebooks. PDAs werden meist nur kurz, of nur für Sekunden, eingeschaltet, um bestimmte Informationen abzufragen oder einzugeben. Daher dürfen PDAs keine langen Hochfahrzeiten haben, sondern müssen nahezu verzögerungsfrei bedient werden können.

Neben den Universalgeräten gibt es auch Geräte mit festem Programm. Der Begriff Handheld wird für derartige Geräte meist nicht mehr verwendet, stattdessen richtet sich der Name nach der Funktion des Spezialgerätes:

- PDAs am ähnlichsten sind elektronische Kalender mit festem Programm. Sie haben einen sehr reduzierten Funktionsumfang und erlauben oft keinen Abgleich mit stationären Rechnern.
- Elektronische Lesestifte sind kleine Scanner. Es werden entweder Strichcodes oder gedruckte Texte eingelesen um diese elektronisch weiter zu verarbeiten.
- E-Books dienen zur Speicherung und Wiedergabe elektronischer Bücher.
- Web-Pads dienen zum Laden und Darstellen von Seiten aus dem World Wide Web. In diese Geräte ist ein Web-Browser fest integriert.
- Handhelds mit fester Programmierung werden auch zur mobilen Datenerfassung, z.B. in der Lagerhaltung, eingesetzt.
- GPS-Empfänger: Als Handheld-Gerät besitzen diese neben der Hard- und Software zur Positionsbestimmung oft Karten- und Navigationsfunktionen.

Was das Betriebssystem von Handhelds (sowohl Universal- als auch Spezialgeräte) betrifft, so haben sich zwei Hauptströmungen manifestiert. Die meisten Handhelds basieren auf PalmOS der Firma Palm Inc. oder Windows CE bzw. dessen Nachfolgesystem Pocket PC von Microsoft.

2.1.2 Drahtlose Netzwerke

Die Vorteile drahtloser Netze – von *Personal Area Network* (PAN) bis *Local Area Network* (LAN) – sind leicht zu erklären. Der größte Vorteil liegt in der Mobilität der Benutzer des Netzes. Auch das Kabelgewirr mit Problemen wie fehlerhafte Kabeln oder inkompatible Steckverbindungen entfällt bei drahtlosen Netzen zur Gänze.

Da im vorliegenden Praktikum *Wireless Local Area Network* (WLAN) zum Einsatz kommt, wird diese Technik als erstes und am genauesten beschrieben. Anschließend werden noch ein paar, mehr oder weniger erfolgreiche, Alternativen genannt und kurz erläutert.

WIRELESS LOCAL AREA NETWORK (WLAN).

Der Begriff WLAN wird zumeist auf zwei verschiedene Arten verwendet. Zum einen als Sammelbegriff für alle drahtlosen lokalen Netzwerke, zum anderen steht WLAN für drahtlose Netze, die auf dem Standard IEEE 802.11 aufbauen.

Die Spezifikation des WLAN-Standards nach IEEE 802.11 startete 1997. Die erste Spezifikation sah Datenraten bis 2 Mbit/s vor. Im Jahr 1999 wurden die Erweiterungen 802.11a mit maximal 54 Mbit/s und 802.11b mit maximal 11 Mbit/s vorgestellt. Der

Standard 802.11 ist nur einer von vielen IEEE-802-Netzwerkspezifikationen und passt sich in das Gerüst weiterer Spezifikationen ein. Die aus anderen IEEE-802-Standards bekannte Einteilung in die Schichten Logical Link Control, Media Access Control und Physical Layer wurden in den Standard 802.11 übernommen (siehe Tabelle 2, vgl Roth, 2002 und Kafka, 2005).

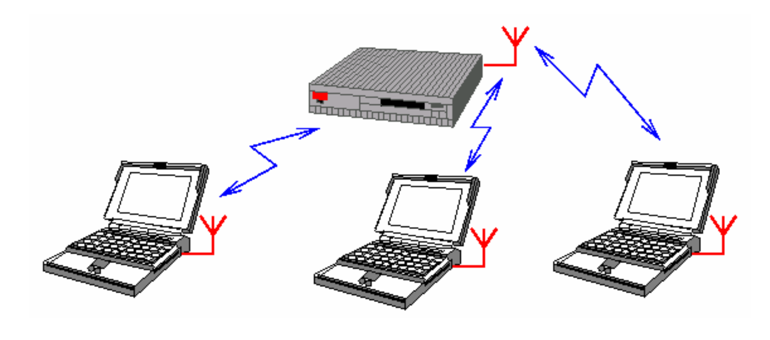
Tabelle 2: Die Protokollarchitektur von IEEE 802.11

Sicherungsschicht (Data Link Layer)	802.2 Logical Link Control		
	802.11 MAC (Media Access Control)		
Bitübertragungsschicht (Physical Layer)	802.11 PLCP (Physical Layer Convergence Protocol)		
	802.11 PMD Infrarot	802.11 PMD FHSS (Frequency Hopping Spread Spectrum)	802.11 PMD DSSS (Direct Sequence Spread Spectrum)

Quelle: Roth, 2002

Ein WLAN nach 802.11 kann in zwei verschiedenen Modi betrieben werden. Im *Infrastruktur-Modus* (siehe Abbildung 1) werden mobile Rechner (Stationen) grundsätzlich über feste Basisstationen (Access Points) angebunden. Diese Access Points sind Rechner, die sowohl über eine drahtlose Anbindung gemäß IEEE 802.11 als auch über eine drahtgebundene Anbindung z.B. nach IEEE 802.3 verfügen. Damit bieten Access Points den Stationen Zugang in ein stationäres Netzwerk. Dieses drahtgebundene Netzwerk dient auch den Access Points zum Austausch von Informationen, wenn die Stationen zwischen den Funkzellen wandern. Als Access Points werden in der Regel Spezialgeräte eingesetzt.

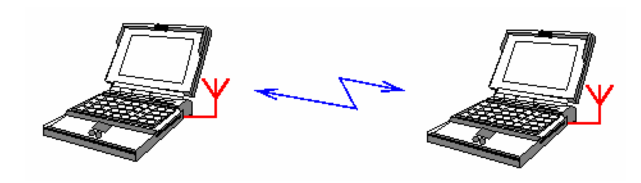
Abbildung 1: WLAN Infrastruktur-Modus



Quelle: http://rnvs.informatik.tu-chemnitz.de/wlan/modi.htm

Im *Ad-hoc-Modus* (siehe Abbildung 2) werden mobile Rechner nur untereinander, ohne Access Point, verbunden. Es erfolgt somit keine Anbindung an ein stationäres Netz. Wird kein höheres Protokoll zur Weitergabe von Paketen eingesetzt, so können nur Stationen kommunizieren, die sich in gegenseitiger Kommunikationsreichweite befinden.

Abbildung 2: WLAN Ad-hoc Modus

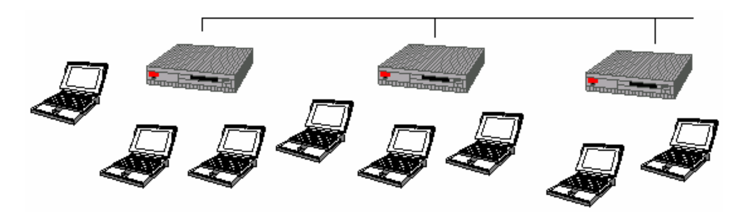


Quelle: http://rnvs.informatik.tu-chemnitz.de/wlan/modi.htm

Der Hauptvorteil des Infrastruktur-Modus ist, dass der Access Point bestimmte zentrale Funktionen zur Koordination anbieten kann. Beispielsweise erfolgt die Synchronisation der Uhren oder auch das Power Management über den Access Point. Im Ad-Hoc-Modus dagegen, sind alle Stationen gleichberechtigt.

Als wichtige Entität eines WLAN kennt IEEE 802.11 den Begriff der *Service Sets*. Ein *Basic Service Set* (BSS) liegt dann vor, wenn zwei oder mehrere Rechner miteinander verbunden sind. Einer davon kann ein Access Point sein. Das einfachste BSS ist das *Independent Basic Service Set* (IBSS). Es entsteht, wenn zwei oder mehr Stationen im Ad-hoc-Modus miteinander verbunden sind und kein Access Point beteiligt ist. Sind mehrere BSS miteinander verbunden, spricht man von einem *Extended Service Set* (ESS). Die Hauptidee dabei ist, dass es sich aus der Sicht der Logical Link Control (LLC) nicht von einem IBSS unterscheidet. Um diese Transparenz zu erhalten, müssen die Access Points über ein so genanntes *Distribution System* (DS) verbunden werden. IEEE 802.11 legt nicht fest, um was es sich genau bei einem DS handelt. Es kann selbst wieder aus drahtlosen oder stationären Netzwerken bestehen, die über Brücken verbunden sind. Ein ESS bildet die Voraussetzung für den automatischen Zellenwechsel mobiler Stationen, das so genannte *Roaming* (siehe Abbildung 3).

Abbildung 3: Infrastruktur-Modus mit mehreren Access Points



Quelle: http://rnvs.informatik.tu-chemnitz.de/wlan/modi.htm

Die letzte Komponente der 802.11-Architektur ist das *Portal*. Es stellt eine Verbindung zu weiteren Netzwerken dar. Hierbei handelt es sich eher um eine logische Komponente als um konkrete Hardware. So kann auch ein Access Point die Aufgabe eines Portals übernehmen, wenn dadurch ein Durchgriff in ein stationäres Netzwerk ermöglicht wird.

Zur Erweiterung von IEEE 802.11 wurden Arbeitsgruppen gebildet, die für die Spezifizierung weiterer Standards zuständig sind. Einen beispielhaften Auszug liefert Tabelle 3.

Tabelle 3: Die IEEE-802.11-Standards und -Arbeitsgruppen

IEEE-Standard/Gruppe	Beschreibung
802.11	WLAN für 1-2 Mbit/s auf dem 2,4-GHz-Band
802.11a	WLAN bis 54 Bit/s auf dem 5-GHz-Band
802.11b	Erweiterung von 802.11 bis 11 Mbit/s auf dem 2,4-GHz-Band
802.11b-cor	Korrekturen zu 802.11b
802.11d	Anpassung an nationale Regelungen
802.11e	MAC-Erweiterung zu 802.11a und b, um Quality of Service und verbessertes Power Management zu ermöglichen
802.11f	Kommunikation zwischen Access Points (IAPP, Inter Access Point Protocol)
802.11g	Höhere Datenraten (ab 20Mbit/s) auf dem 2,4-GHz-Band
802.11h	Höhere Datenraten auf dem 5-GHz-Band
802.11i	MAC-Erweiterung, um verbesserte Sicherheits- und Authentifikationsmechanismen zu ermöglichen

Quelle: Roth, 2002

Bis zum heutigen Tag sind noch einige Arbeitsgruppen hinzugekommen, die sich mit der Spezifikation weiterer Details beschäftigen (vgl. Kafka, 2005). Den aktuellsten Status der Fortschritte der einzelnen Arbeitsgruppen des WLAN-Standards findet man im Internet unter http://grouper.ieee.org/groups/802/11/.

Ein viel diskutiertes Thema in Zusammenhang mit WLAN, ist das Thema Sicherheit. Abgesehen von etwaigen Problemen technischer Natur, werden von vielen Unternehmen nicht einmal die einfachsten Sicherheitsvorkehrungen getroffen. Die Ergebnisse einer im März 2005 publizierten RSA-Security-Studie sind erschreckend: Getestet wurden drahtlose Netzwerke von Unternehmen in Paris, London, San Francisco, New York und Frankfurt.

Frankfurt schnitt dabei am besten ab. Hier fand man „nur“ 34% der Unternehmensnetze nicht ausreichend geschützt, und „nur“ 30%, bei denen nicht einmal die minimalen Sicherheitseinstellungen aktiviert waren. Gemäß der diesjährigen Studie „Computer Crime and Security Survey“ des FBI sind die traditionellen Attacken rückläufig, während Vorfälle im Bereich WLAN zunehmen (vgl. Serloth, 2005). Viele der Attacken können durch die Beachtung folgender elementarer Sicherheitseinstellungen vermieden werden:

- Name des Netzwerkes wählen, ohne Rückschlüsse auf das Unternehmen zu ermöglichen.
- Ad-hoc-Modus deaktivieren, um den Zugriff auf die Festplatteninhalte durch Außenstehende zu verhindern.
- Zugriffsmöglichkeit nur für vorgegebene MAC-Adressen.
- Ändern eventuell voreingestellter Netzwerkkennungen und Passwörter.
- Verschlüsselungseinstellungen der Daten laut Benutzerhandbuch.

HIPERLAN

Der HiperLAN/1-Standard für schnelle LAN-Anwendungen wurde bereits im Jahr 1997 verabschiedet. Es handelt sich um einen Funk-LAN-Standard im Frequenzbereich 5 GHz für die Hochgeschwindigkeitskommunikation zwischen tragbaren Geräten. Mit diesem Standard ist es möglich drahtlose, multimediafähige Datennetze mit Übertragungsraten bis zu 20 Mbit/s zu realisieren. Die funktionalen Eigenschaften sind in der europäischen Norm EN 300 652 spezifiziert. Trotz der frühen Verabschiedung des Standards sind heute keine Produkte am Markt verfügbar.

Mit HiperLAN/2 soll den Benutzern sowohl in geschäftlichen als auch in öffentlichen oder privaten Bereichen ein Breitband-Internetzugang mit definierten Qualitätsmerkmalen bereitgestellt werden. Die Übertragungsraten bis zu 54 Mbit/s und das Zusammenarbeiten mit anderen Technologien wie Ethernet, IEEE 1394 und ATM erlauben auch Videoanwendungen in Echtzeit. Ein bereits im September 2000 gegründetes globales Industrieforum (Bosch, Dell, Ericsson, Nokia, Telia und Texas Instruments) soll für die nötige Marktakzeptanz sorgen (vgl. Kafka, 2005).

HOME RF.

Die Aktivitäten der HomeRF Working Group (HRFWG), die im März 1998 begannen, waren ganz auf die Bedürfnisse von privaten Anwendern ausgerichtet. Basis für die drahtlose Kommunikation innerhalb privater Wohnungen und Häuser ist SWAP (*Shared Wireless Access Protocol*). Die erzielten Datenraten liegen zwischen 1 und 2 Mbit/s. Eine schon geplante Breitband-Version sollte Datenraten bis 25 Mbit/s bereitstellen. Obwohl Hersteller wie Proxim und Siemens marktreife Produkte herstellten, musste die HRFWG ihre Aktivitäten, aufgrund der rasanten Entwicklungen der WLAN-Technologie, im Jänner 2003 einstellen (vgl. Kafka, 2005).

BLUETOOTH.

Bluetooth ist ein Funksystem, das zur Übertragung von Sprache und Daten geeignet ist. Ziel der Konzeption ist der Ersatz von Kabel und Infrarot. Bluetooth arbeitet im 2,4 GHz ISM-Band und hat eine typische Reichweite von zehn Meter. Aufgrund der geringeren Reichweite spricht man auch von *Wireless Personal Area Network* (WPAN). Bluetooth zielt weniger auf den betrieblichen als auf den privaten Gebrauch ab.

2.2 Handhelds in der betrieblichen Informationsverarbeitung

In der betrieblichen Informationsverarbeitung kommen Handhelds, bzw. mobile Endgeräte allgemein, in erster Linie im Bereich der mobilen Datenerfassung zum Einsatz. Anwendungen dieser Art kommen überall dort in Frage, wo Daten genau, zuverlässig und vor allem schnell und mobil aufgenommen werden sollen bzw. müssen. Durch die Vielzahl an Erweiterungsmodulen, wie zum Beispiel Strichcode-Identifikation oder Positionsbestimmung mittels GPS (*Global Positioning System*), werden mobile Endgeräte zu universellen Erfassungs- und Verarbeitungsgeräten und finden in vielen Bereichen Verwendung (vgl. Marlovits, 2002).

2.2.1 Verbreitung und Nutzen

Generell kann man sagen, dass die Verwendung von Handhelds als Teil der betrieblichen Informationstechnologie, einen Aspekt der Themen *Mobile Business* und *M-Commerce* darstellt. In diesem Subkapitel wird nur auf die betrieblichen Bereiche mit den größten Mobilitätspotenzialen eingegangen. Darüber hinaus ist der betriebliche Einsatz von Handhelds auch in anderen Bereichen sinnvoll und nur durch den Einfallsreichtum des verantwortlichen Entscheidungsträgers im Unternehmen beschränkt.

MATERIALWIRTSCHAFT.

Der hauptsächliche Anwendungsbereich im Zuge der Materialwirtschaft ist die Inventarisierung. Da dieser Prozess oft eine langwierige manuelle Bearbeitung von Listen in Form von Datenbankausdrucken erfordert, ist in diesem Bereich eine enorme Effizienzsteigerung durch die Verwendung von mobilen Geräten zur Datenerfassung möglich. Um den Arbeitsprozess zu optimieren kommen hauptsächlich spezielle Handhelds mit eingebauten Scannermodulen zum Einsatz, mit denen die Inventur rascher und weniger fehleranfällig durchgeführt werden kann. Auch eine, im Normalfall noch schnellere, Bedienung durch Sprachsteuerung, wie im vorliegenden Praktikum, ist möglich. Die Datensynchronisation erfolgt entweder offline oder online. Im ersten Fall werden die Daten durch eine *Middleware* gesammelt und in periodischen Läufen an die Applikation übertragen. Erfolgt die Datenerfassung in Echtzeit, werden die Daten direkt über drahtlose Funknetze an die Applikation geleitet und stehen sofort zur weiteren Verarbeitung zur

Verfügung. Der Vorteil dabei liegt in der Möglichkeit, die Inventur ohne Probleme mit mehreren Geräten durchzuführen (vgl. Steimer, 2001)

Darüber hinaus lassen sich Handhelds natürlich auch noch in anderen Bereichen der Materialwirtschaft erfolgreich einsetzen. Die Flexus AG hat bereits bei mehreren Kunden mobile Scanner im Bereich der Kommissionierung erfolgreich eingeführt. Sowohl im Bereich der Kundenauftragsabwicklung (Kommissionierung zum Kundenauftrag) als auch in der Produktion (Kommissionierung zum Fertigungsauftrag) konnten dadurch erhebliche Zeit- und somit auch Kosteneinsparungen erzielt werden.

Weiters kommen mobile Datenerfassungsgeräte auch bei der Warenannahme zum Einsatz. Die Verbuchung des Wareneingangs zur Lieferantenbestellung kann, durch die Nutzung von Strichcodes und dementsprechende Lesegeräte, schnell und fehlerfrei durchgeführt werden, ohne dass der betroffene Sachbearbeiter die Abladestelle verlassen und zu einem stationären Terminal gehen muss.

VERTRIEB.

Zahlreiche Unternehmen setzen für die Akquisition von Aufträgen und Verkaufsberatung im Außendienst wirkende Mitarbeiter ein (*Mobile Sales*). Vielfach müssen gerade bei derartigen Verkaufsgesprächen vor Ort Informationen von einer entfernt befindlichen Institution oder der eigenen Firmenzentrale eingeholt werden. Diese Informationen können rasch über einen Handheld, welcher zumeist über Mobilfunk mit dem gewünschten System verbunden wird, eingeholt und dargestellt werden. Folgende Funktionen können über einen Handheld durchgeführt werden (vgl. Steimer, 2001):

- ✍ Abruf aktueller Produktinformationen durch den Vertriebsbeauftragten.
- ✍ Abruf kundenrelevanter Informationen für das Verkaufsgespräch.
- ✍ Einholung einer verkaufsrelevanten Entscheidung (z.B. Informationen zu Sonderkonditionen).
- ✍ Einholung von Informationen zur freieren Planung der Verkaufstour.

Nicht zu vergessen ist auch der mögliche Einsatz des Handhelds als Navigationsgerät für mobile Außendienstmitarbeiter, sofern das Gerät mit einem optionalen GPS-Empfänger ausgestattet ist. Dadurch können unnötige Lehrlaufzeiten, die durch das Suchen der benötigten Adressen entstehen, vermieden werden.

PERSONAL INFORMATION MANAGEMENT.

In allen Unternehmensbereichen können Handhelds zur effizienten Gestaltung des Informationsflusses genutzt werden. In erster Linie liegt der Fokus der Hersteller, entsprechend den Anwenderprioritäten, auf mobilen Mail-Anwendungen. Erst danach werden Themen wie Personal Information Management, Kalender-Anwendungen, CRM-Lösungen oder Sales Force Automation behandelt. Grund dafür ist unter anderem der Markerfolg des kanadischen Unternehmens Research in Motion (RIM) und ihr Produkt

BlackBerry. Heute kann der Benutzer bereits unter mehreren Alternativen wählen (vgl. Giordano, 2005).

Abschließend kann gesagt werden, dass die Investitionsbereitschaft in mobile Unternehmensanwendungen hoch ist. Im wachsenden Markt der Unternehmensmobilität setzen beispielsweise 58% der US-amerikanischen „Mobile Enterprise“ mehr als 500 Geräte ein, 16% der Unternehmen investierten im vergangenen Jahr mindestens $500.000 in Endgeräte. Genutzt und geplant sind Applikationen in allen Bereichen des unternehmerischen Informationsaustausches, wobei sich, entsprechend der Breite der Branchen und möglichen Anwendungsfelder, kein „Killerbereich“ herauskristallisiert (vgl. Giordano, 2005).

2.2.2 Case Studies aus Österreich

Neben zahlreichen Projekten in Deutschland konnte die Firma *Flexus AG* auch einige Kunden in Österreich gewinnen. Einer davon ist der Sportartikelhersteller Atomic. Die Firma Atomic hat das gesamte Datenmanagement in der Lagerwirtschaft des Standortes Altenmarkt auf eine mobile Funkscanner-Lösung umgestellt. Betroffen waren sämtliche Prozesse, vom Wareneingang bis zur Auslagerung an die Produktion. Technische Basis dieser Lösung ist *SAP Logistics Execution System* (LES), ein Bereich von *mySAP Supply Chain Management*. Zur mobilen Datenerfassung kommen Handhelds der Firma *Intermec* mit integrierten Scanner-Modulen zum Einsatz.

Das Projekt war Teil einer Optimierungsoffensive zur Steuerung eines schnellen, effizienten und strukturierten Warenumschlags im Materiallager. Seit erfolgreicher Beendigung dieses Projekts, haben sich die Lagerabläufe bei Atomic deutlich verbessert und an Qualität gewonnen. Das beginnt beim Wareneingang, wo die Daten der angelieferten Waren über Barcode-Labels eingescannt, online auf Richtigkeit geprüft und für den weiteren Prozess vorbereitet werden. Fehlerhafte Lieferungen lassen sich bereits zu diesem Zeitpunkt erkennen und gelangen nicht in den Bearbeitungskreislauf. Beim Einlagern von Material wird ebenfalls der Barcode gescannt und somit dem Backendsystem der betreffende Lagerplatz übermittelt. Beim Warenausgang verhält es sich im Prinzip genauso. Lediglich die Transportnummer (aus dem SAP-System) wird eingescannt. Relevante Daten wie Materialnummer, Menge und Lagerplatz werden online bereitgestellt. Der Vollzug der Auslagerung (First in – First out) wird mittels Barcodescannung des Lagerplatzes gemeldet. Durch die Onlineübertragung aller Warenbewegungen entfallen viele zeitintensive Teilschritte. Die Mitarbeiter werden von operativen Tätigkeiten entlastet. Laut Aussage des zuständigen Leiters für Einkauf und Arbeitsvorbereitung, sorgen der papierlose Ablauf und die konkrete Lokalisierung der Materialbestände auf Lagerplatzebene für mehr Qualität, Geschwindigkeit und Produktivität (vgl. SAP AG, 2002a).

Neben der Flexus AG gibt es auch noch weitere Dienstleistungsunternehmen in Österreich, die sich im Bereich mobiler Datenerfassung spezialisiert haben, und sowohl Hardware als auch die benötigte Beratung zur Einführung funktionierender Systeme bereitstellen.

Ein Beispiel eines solchen Unternehmens ist die *Bruck Technologies Datascan International Gruppe* mit Stammsitz in Wien. Bruck Technologies ist Komplettanbieter für Automatische Datenerfassungssysteme mit mehr als 100 Mitarbeitern in zehn Ländern in Zentral- und Osteuropa. Das Unternehmen bietet Produkte, Lösungen und Services für innovative Logistiksysteme in vielen Unternehmensbereichen in Handel, Gewerbe und Industrie. Basierend auf den Identifikationstechnologien Barcode und RFID werden die Kunden beginnend von der Planung und Projektierung bis hin zur Installation und Implementierung bzw. Anbindung an vorhandene Softwaresysteme, wie SAP und andere ERP Systeme beraten. Am Beispiel des traditionsreichen Kaffeeproduzenten Santora zeigt sich der Nutzen einer modernen Logistik-Lösung mit Handhelds und WLAN-Koppelung für den Außendienst (vgl. Bruck Technologies, o.J.).

Als Ersatz für eine bestehende Lösung zur mobilen Datenerfassung entschloss man sich bei Santora Kaffee 2001, die Fahrverkäufer mit modernen Handhelds auszustatten, die ihnen auf Knopfdruck ein Maximum an Informationen für ihre tägliche Arbeit zur Verfügung stellen. Das eingesetzte System wurde mit Hardware-Komponenten von *Symbol Technologies* von der mainwork information technology AG, einem Partner der Bruck Technologies, als Generalunternehmer verwirklicht. Die bei Santora eingesetzte Software stammt ebenfalls von der mainwork information technology AG.

Die von Symbol Technologies gelieferten Handheld-Computer sind mit einer WLAN-Karte ausgestattet, die es jedem, mit dem Gerät ausgestatteten Fahrverkäufer, vor Beginn seiner wöchentlichen Tour ermöglicht, sämtliche relevanten Informationen vom Zentralrechner, einer IBM AS/400 (IBM iSeries), herunter zu laden. Diese Informationen beinhalten die zu fahrende Route, die anstehenden Aufträge und Zusatzinformationen über den Kunden. Bei der Beladung seines Transporters nutzt der Fahrer den im Handheld integrierten Barcode-Scanner zur Erfassung der Waren und zur sofortigen Aktualisierung des „Fahrzeug-Lagerstandes“. Beim Kunden angekommen, nutzt der Fahrverkäufer seinen Handheld erneut, um ausgegebene Artikel wiederum mittels Barcode-Scanner zu erfassen. Ausgabe und Abgänge von Waren werden automatisch registriert, im Handheld erfolgt eine sofortige Korrektur des Lagerbestandes im Fahrzeug. Der Fahrer hat dadurch ohne großen Aufwand den aktuellen Stand seiner Ware immer im Blick und kann detaillierte Aussagen über verkaufbare Mengen tätigen.

Eine genaue Lagerführung kann am Ende der Tour auf Knopfdruck abgerufen werden. Rechnungen und Lieferungsbestätigungen werden beim Kunden vor Ort über einen mobilen Drucker ausgedruckt. Es entfällt die fehleranfällige und aufwendige Ausstellung per Hand. Ebenso ist eine Inkassofunktion in der Software verfügbar: Außenstände können inkassiert und sofort verbucht werden. Am Ende seiner wöchentlichen Tour ermöglicht es das System dem Fahrer, sämtliche erfasste Daten, Rechnungen, Aufträge, Inkassoinformationen, Lagerabgänge, etc., per Knopfdruck automatisch über das integrierte WLAN mit dem zentralen Host zu synchronisieren. Diese Datenübermittlung kann für die Handhelds aller Fahrer gleichzeitig erfolgen.

2.3 Ausgewählte Anbindungsmöglichkeiten von Handhelds an SAP

Die SAP AG verfolgt die Idee der *Enterprise Service Architecture* (*ESA*) und stellt mit dem *SAP NetWeaver* eine offene Integrations- und Anwendungsplattform zur Verfügung. So können Geschäftsprozesse über Technologiegrenzen hinweg vereinheitlicht werden und verschiedene Anwendungen nach Bedarf integriert werden. Dabei soll einfach und strukturiert auf Systeminformationen zugegriffen werden können.

Zur Anbindung von Handhelds an SAP Systeme werden von der Flexus AG verschiedene Möglichkeiten vorgeschlagen. Nur jene Möglichkeiten, die auch von der SAP AG offiziell unterstützt werden kommen in Betracht. Abbildung 4 zeigt einen Überblick über folgende Lösungsansätze:

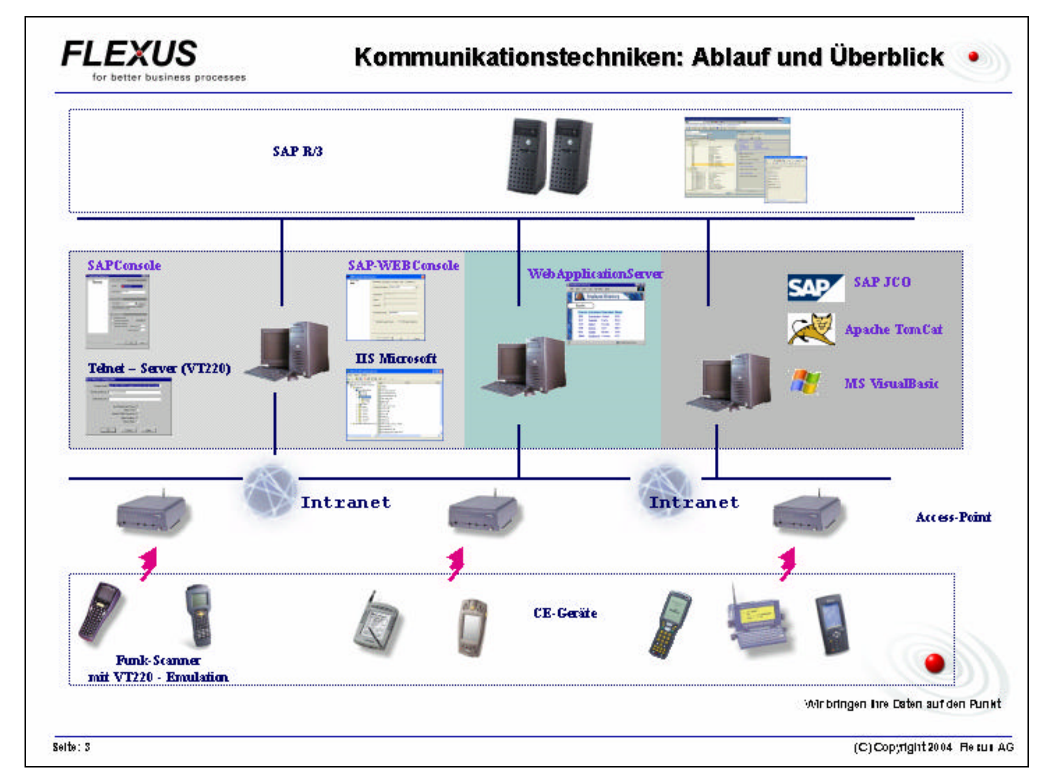
i. SAP Console

ii. SAP WEBConsole

iii. SAP Web Application Server (WAS)

iv. SAP Java Connector (JCO)

Abbildung 4: Kommunikationstechniken: Ablauf und Überblick



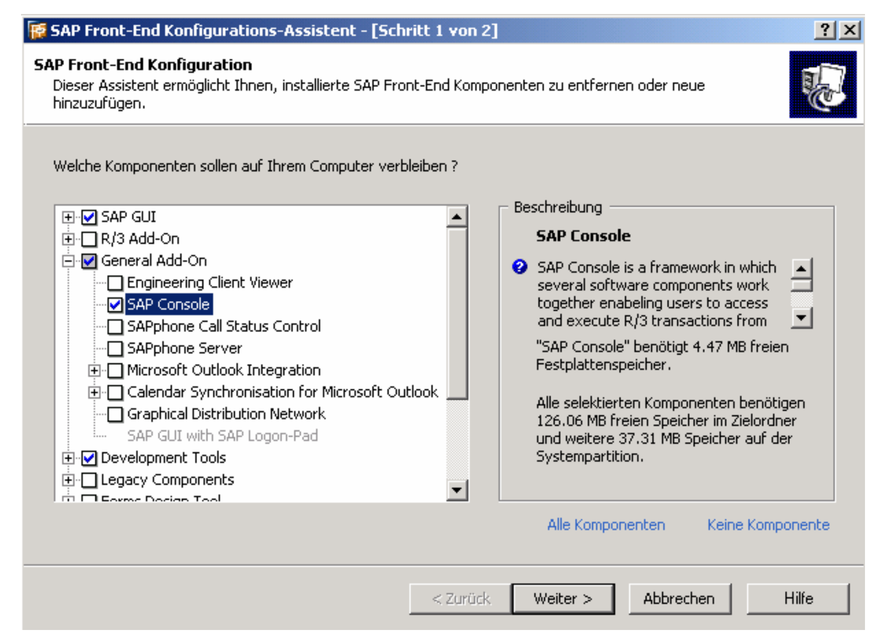
Quelle: Flexus AG (2004)

Beim SAP Java Connector handelt es sich um ein Toolkit mit dessen Hilfe eine beliebige Java Anwendung mit einem SAP System kommunizieren kann. Da für diese Variante keine integrierte SAP Entwicklungsumgebung genutzt wird und es sich somit nicht um eine SAP-spezifische Lösung handelt, wird im Folgenden nur auf die SAP Console bzw. SAP WEBConsole und auf den SAP Web Application Server eingegangen.

2.3.1 *Lösung mittels SAP Console*

Mit dieser Lösung bietet die SAP eine Möglichkeit zur mobilen Datenerfassung z. B. mittels Handscannern. Diese werden über direkten Datenfunk in das Netz eingebunden. Die mobilen Funkterminals empfangen die Daten direkt aus dem SAP System und übertragen Ergebnisdaten zurück. Die SAP Console wird als Komponente des SAP GUI installiert (vgl. Abbildung 5).

Abbildung 5: SAP Front-End Konfiguration



Quelle: Eigene Abbildung

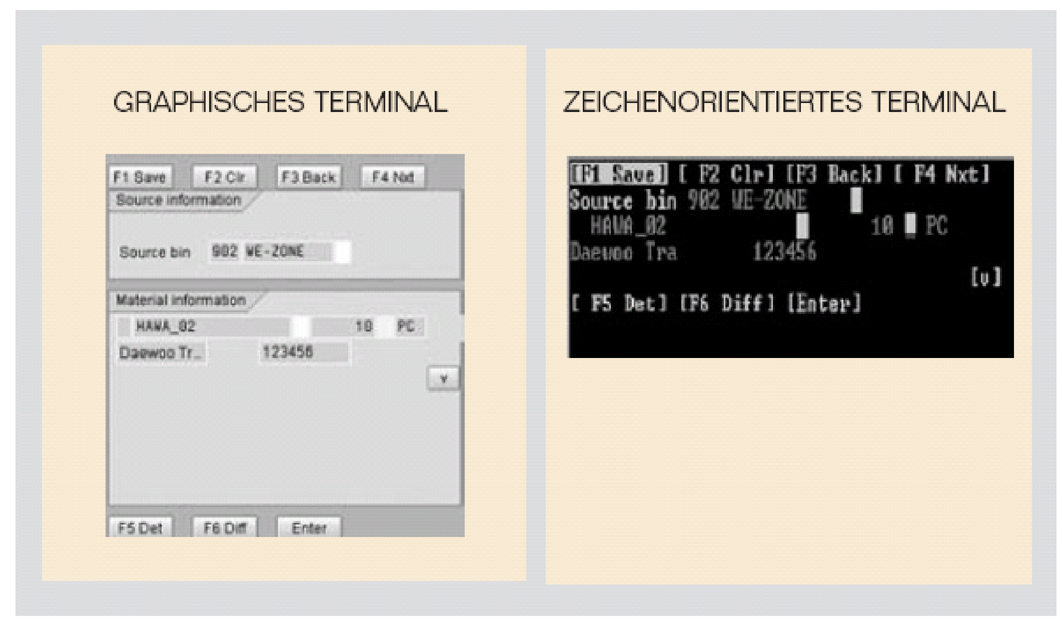
Nach erfolgreicher Installation werden die benötigten Einstellungen mittels integriertem *SAPConsole Administrator*, der in der Systemsteuerung des Betriebssystems angelegt wird, vorgenommen. Es wird festgelegt an welchem System das mobile Terminal angemeldet und welche Transaktion gestartet werden soll (vgl. Abbildung 6).

Abbildung 6: SAP Console Administrator – allgemeine Einstellungen

Quelle: Eigene Abbildung

Sofern die Anforderungen des Unternehmens nicht durch ausgelieferte Transaktionen abgedeckt werden, können eigene Transaktionen in der integrierten Entwicklungsumgebung (ABAP Development Workbench) des SAP Systems programmiert werden. Diese Transaktionen können im SAP System wie gewohnt über das *SAP GUI* (Graphical User Interface) aufgerufen und getestet werden. Wird mit einem zeichenorientierten Terminal gearbeitet werden die Transaktionen online über die SAP Console auf eine zeichenorientierte ASCII-Benutzeroberfläche umgesetzt (vgl. SAP AG, 2002b). Abbildung 7 zeigt die Aufbereitung je nach Benutzeroberfläche.

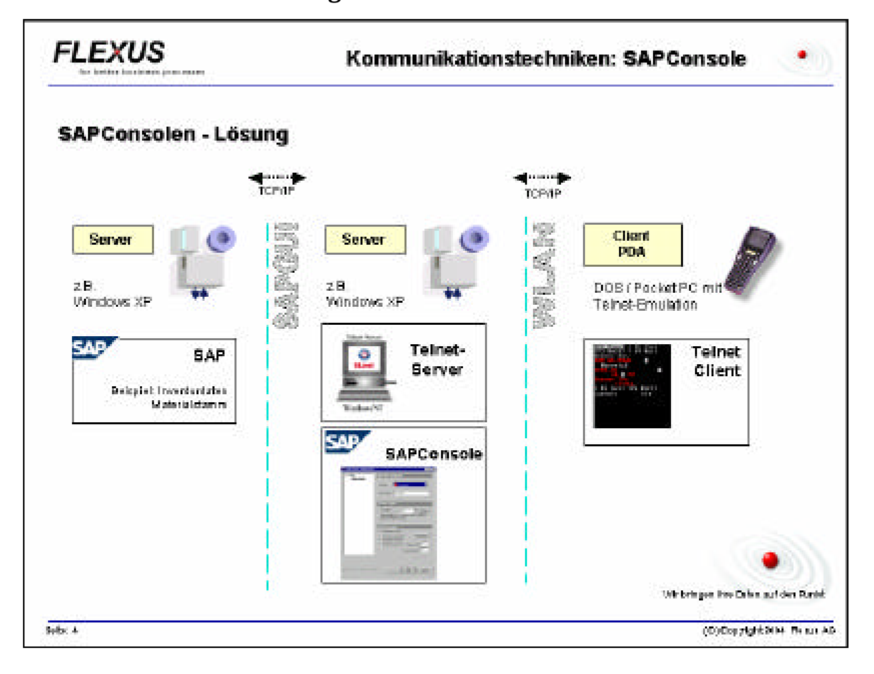
Abbildung 7: Direkter Datenfunk



Quelle: SAP AG (2002)

Die Netzanbindung des mobilen Terminals erfolgt über WLAN. Über ein Script, das die Zieladresse des Servers und etwaige Anmeldedaten enthält, wird ein Telnet-Client gestartet und die Verbindung mit dem Telnet-Server hergestellt. Der Telnet-Server entscheidet anhand des Users welches Programm (in diesem Fall die SAP Console) gestartet werden soll. Wie bereits oben erwähnt, wird zuvor über den SAP Console Administrator festgelegt welches SAP System und welche Transaktion anschließend gestartet wird (vgl. Abbildung 8).

Abbildung 8: Datenfluss SAP Console

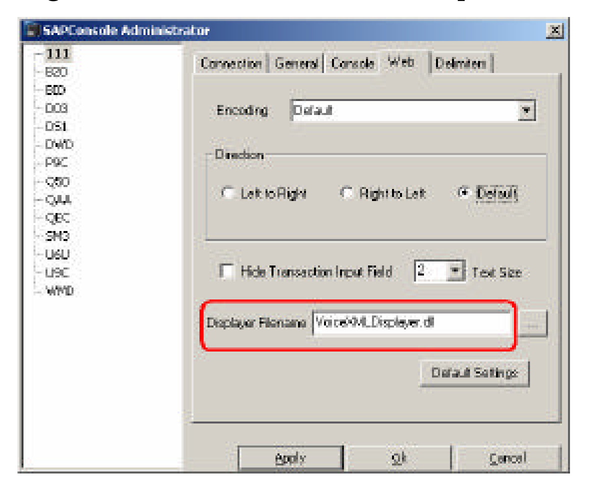


Quelle: Flexus AG (2004)

2.3.2 Lösung mittels SAP WEBConsole

Hierbei handelt es sich um die neueste Version der SAP Console. Die Scanner-Informationen können bei graphischen Terminals mit Windows CE bzw. Windows Pocket PC auf einem Webbrowser dargestellt werden. Über die Weberweiterung besteht nun die Möglichkeit neue Technologien wie Spracherkennung einzubinden (vgl. SAP AG, 2002). Um diese Möglichkeit zu nutzen muss der richtige Displayer gewählt werden (vgl. Abbildung 9).

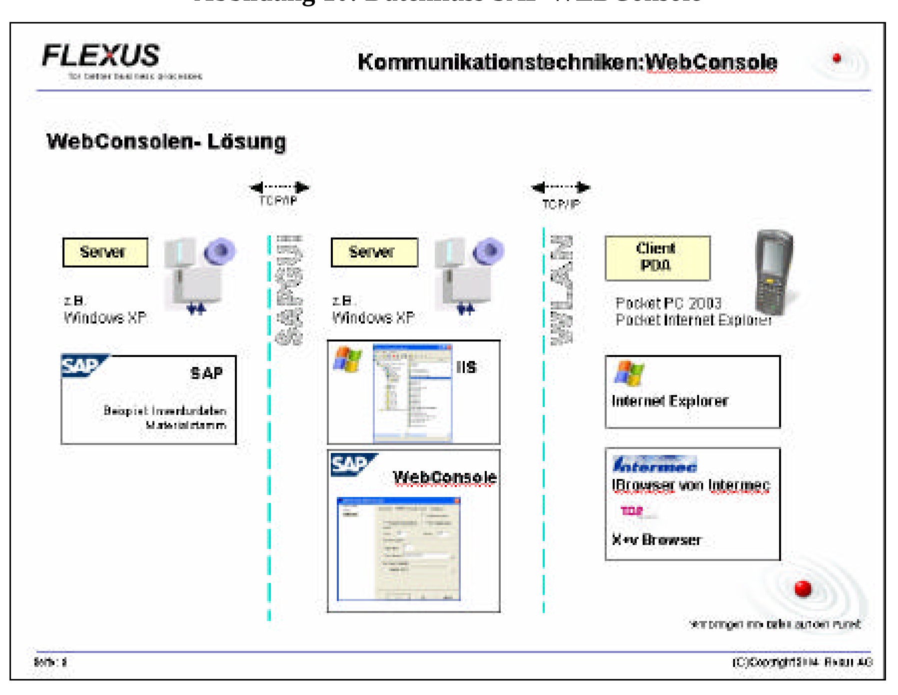
Abbildung 9: SAP Console Administrator – Spracherkennung



Quelle: SAP AG (2004)

Der Datenfluss verläuft analog zur früheren SAP Console. Der einzige Unterschied besteht darin, dass statt einem Telnet-Server Microsoft IIS (Internet Informationsdienste) zur Kommunikation genutzt werden (vgl. Abbildung 10).

Abbildung 10: Datenfluss SAP WEBConsole



Quelle: Flexus AG (2004)

2.3.3 Lösung mittels SAP Web Application Server

Der Web Application Server (WAS) stellt die technologische Weiterentwicklung des bisherigen SAP Basissystems dar. Er ist die zugrunde liegende Technologie für fast alle SAP Lösungen. Neben dem SAP Programmiermodell (Development Workbench) bietet der WAS:

i. Native Unterstützung von *HTTP*, *HTTPS*, *SMTP* und anderen Internetprotokollen

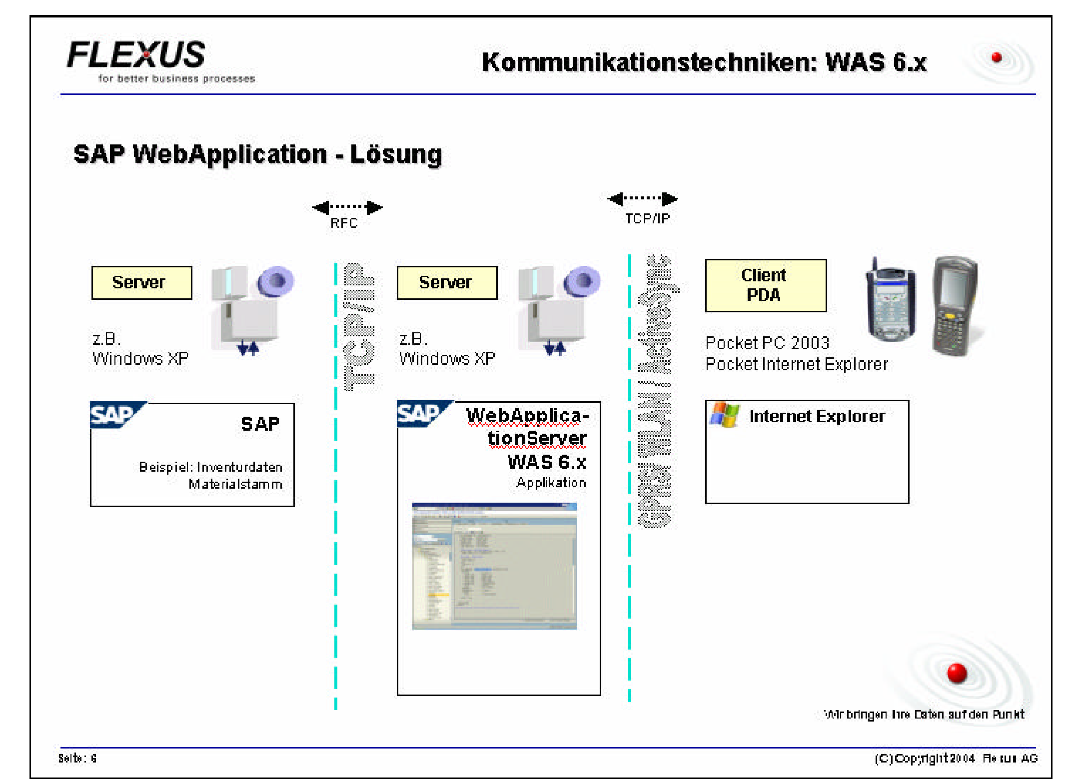
ii. Serverseitiges Scripting in ABAP und JavaScript

iii. Eine in die Development Workbench integrierte Entwicklungsumgebung zur Erstellung von Webanwendungen

Mit dem WAS können Internetanwendungen (z.B. Internet Shops) schnell und einfach entwickelt werden. Die Darstellung der einzelnen Anwendungen ist in Business Server Pages (BSP) organisiert (vgl. SAP Online Documentation, 2005a).

Der WAS holt die benötigten Business-Daten aus dem SAP System und stellt dem Browser auf dem Handheld die automatisch generierten HTML Seiten zur Verfügung. Abbildung 11 zeigt den Datenfluss vom Backendsystem über den WAS zum Browser.

Abbildung 11: Datenfluss SAP Web Application Server



Quelle: Flexus AG (2004)

2.3.4 *Gegenüberstellung der ausgewählten Anbindungsmöglichkeiten*

Da die Vor- und Nachteile der SAP WEBConsole jene der SAP Console beinhalten wird auf letztere in der Gegenüberstellung nicht mehr extra eingegangen (siehe Tabelle 4).

Tabelle 4: Gegenüberstellung SAP WEBConsole und SAP Web Application Server

Bereich	WEBConsole		Web Application Server	
Lizenz	☺	Bestandteil des SAP Front-End		Abhängig von der Lizenzpolitik der SAP AG
Benötigte Zusatz-Software		Telnet-Server bzw. Microsoft IIS bei Funkterminals	☺	Keine
Installation	☺	Einfach und schnell		Etwas aufwändig, da komplettes System
Zusatznutzen		Keiner	☺☺	Basis für fast alle SAP Lösungskomponenten (somit meist ohnehin vorhanden)
Lieferumfang		Einige Standardtransaktionen im SAP-System vorhanden	☺	Entwicklungsumgebung und Basiskomponente
Hauptsächliche Nutzung		Logistic Execution System (LES)	☺	Eigenentwicklungen und Vielzahl eigenständige SAP Komponenten (z.B. Online Store)
Know-how Bedarf für Entwicklungen	☺	ABAP Development Workbench		ABAP HTML ev. BHTML
Unterstützung weiterer Sprachen		Nein	☺☺	Java JavaScript Dynamic HTML
Unterstützung TTS und VR	☺	Ja, durch VoiceXMLDisplayer	☺	Ja, durch Integration von VoiceXML im X+V Standard
Freiraum für Eigenentwicklungen		Sehr eingeschränkt im Layout durch automatische	☺☺	Völlige Entwicklungsfreiheit durch Integration

		Generierung der Eingabemasken		verschiedener Programmiersprachen und Techniken
Performance		Gesamte Ablauflogik läuft am Server (direkt im SAP System)	☺	Serverseitiges Scripting mit ABAP oder Java und clientseitiges Scripting mit JavaScript möglich

Quelle: Eigene Tabelle

TTS ....Text to Speech
VR.................Voice Recognition
BHTML.........Business HTML (SAP Auszeichnungssprache – ähnlich HTML)
X+V ....XHTML und Voice (siehe Subkapitel 2.4)

Objektiv betrachtet überwiegen die Vorteile des Web Application Servers. Vor allem bei größeren Unternehmen, die nicht nur die Kernmodule Finance, Controlling und eventuell Material Management des SAP Systems (vormals R/3 System) in Verwendung haben, ist der WAS ohnehin als Basis neuerer SAP Komponenten im Einsatz. Somit liegt es natürlich auch sehr nahe diese mächtige Entwicklungsumgebung für alle benötigten internetgestützten Erweiterungsentwicklungen zu nutzen. Der größte Vorteil liegt in der zentralen Verwaltung aller Webanwendungen sowie deren kontinuierliche Online-Präsenz. Weiters bietet der WAS eine skalierbare und zuverlässige Infrastruktur, die höchste Performance gewährleistet und den internetgestützten Zugriff auf alle unternehmensweiten SAP Komponenten via Browser und einer Reihe mobiler Endgeräte ermöglicht (vgl. SAP Online Documentation, 2005b).

Die SAP WEBConsole sollte man als das sehen, was sie ist. Sie stellt eine unkomplizierte und schnelle Lösung im Logistic Execution System als Teilbereich des Warehouse Managements dar. Im Gegensatz zum WAS, der ein vollständiges Server System ist, ist die WEBConsole ein komfortables und schnell zu installierendes Tool um bestimmte Transaktionen durch den Einsatz von mobilen Funkterminals zu vereinfachen und zu beschleunigen.

Beide Lösungsansätze haben im Bereich von sprachgesteuerten Transaktionen auf Handhelds ihre Daseinsberechtigung. Die Entscheidung zugunsten einer Lösung, im Zuge eines SAP Erweiterungs- oder Einführungsprojektes, kann nur in Zusammenarbeit mit dem jeweiligen Entscheidungsträger im Unternehmen und unter Berücksichtigung aller Begleitumstände, insbesondere der betroffenen Systemlandschaft, getroffen werden.

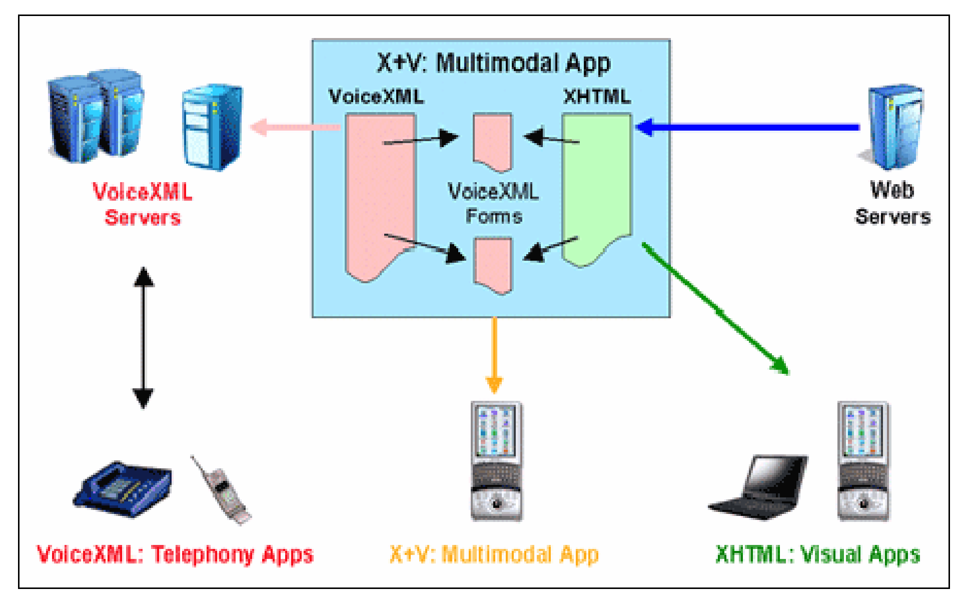
2.4 Sprachausgabe und Spracherkennung

Motorola, Opera Software ASA und IBM unterbreiteten dem World Wide Web Consortium (W3C) einen Vorschlag eines Standards einer Auszeichnungssprache zur Erstellung multimodaler Anwendungen. Dieser Standard wird XHTML + Voice (kurz X+V) genannt und vom W3C in der Spezifikation 1.2 unterstützt (vgl. W3C, 2004a).

Unter einer multimodalen Anwendung versteht man eine Anwendung die dem User ein flexibles Interface liefert, mit dessen Hilfe eine Interaktion aus einer Kombination von Eingabegeräten wie Tastatur, Touch Screen oder Telefontastatur und Sprache, möglich ist. Diese Nutzung der Sprache und deren Kombination mit der visuellen Ausgabe stellt das wichtigste Charakteristikum einer multimodalen Anwendung dar (vgl. IBM, 2003a).

Eine multimodale Applikation setzt sich aus einer visuellen Komponente und einer sprachgesteuerten Komponente zusammen. Die Applikation kann somit von einer Vielzahl von Endgeräten genutzt werden (vgl. Abbildung 12). Technisch erfolgt die Realisierung einer multimodalen Applikation durch die Kombination von XHTML (visuelle Komponente) und VoiceXML (sprachgesteuerte Komponente). Bei beiden Auszeichnungssprachen handelt es sich ebenfalls um offene Standards, die vom W3C unterstützt werden (vgl. W3C, 2004b und W3C, 2005).

Abbildung 12: Komponenten einer multimodalen Applikation



Quelle: IBM (o.J.)

Die *Namespace Declaration* für eine typische X+V Applikation wird mit zusätzlichen Deklarationen für VoiceXML und XML Events in XHTML geschrieben (vgl. Abbildung 13). Bei XML Events handelt es sich ebenfalls um einen offenen Standard der vom W3C unterstützt wird (vgl. W3C, 2003).

Abbildung 13: X+V – Namespace Declaration

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE html PUBLIC "-//W3C/DTD XHTML+Voice 1.0/EN" "xhtml+voice.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:ev="http://www.w3.org/2001/xml-events"
xmlns:vxml="http://www.w3.org/2001/vxml" xml:lang="en_US" >
```

Quelle: IBM (2003b)

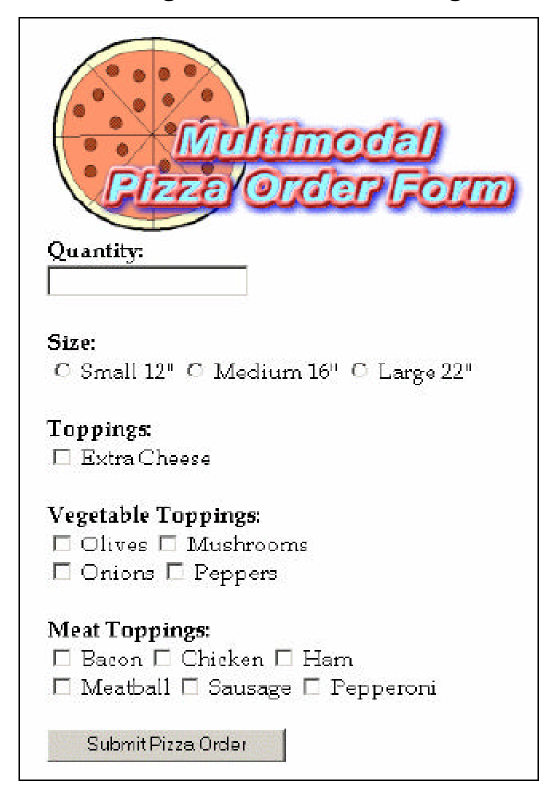
Der visuelle Teil der Seite besteht aus XHTML Code und definiert ein Formular. Dabei können, wie gewohnt, alle Elemente, wie Radio Buttons oder Check Boxes, eines Formulars verwendet werden. Abbildung 14 zeigt einen Ausschnitt eines derartigen Beispiels für ein Formular um Pizza zu bestellen. Abbildung 15 zeigt die dazugehörige Bildschirmausgabe.

Abbildung 14: X+V – Visueller Teil

```
  <b>Size:</b><br/>
  <input type="radio" name="size" id="sizeSmall" ev:event="focus"
ev:handler="#voice_size"/>
      Small 12&quot;
  <input type="radio" name="size" id="sizeMedium" ev:event="focus"
ev:handler="#voice_size"/>
      Medium 16&quot;
  <input type="radio" name="size" id="sizeLarge" ev:event="focus"
ev:handler="#voice_size"/>
      Large 22&quot;
```

Quelle: IBM (2003b)

Abbildung 15: X+V – Visuelle Ausgabe



Quelle: IBM (2003b)

Der Sprachteil der Applikation ist jener Code, der die Sprachaus- und Spracheingabe jedes Formularfeldes steuert. Für die Spracheingabe wird eine spezielle *Grammatik* benötigt, um festzulegen welche Spracheingaben gültig sind. Die Grammatik wird im Normalfall über eine externe Datei eingebunden. Abbildung 16 zeigt den Sprachteil mit dem VoiceXML Code, Abbildung 17 zeigt die dazugehörige Grammatikdatei.

Abbildung 16: X+V – Sprach-Teil

```
<vxml:form id="voice_toppings_vegetable">
    <vxml:field name="vtoppingsvegetable">
      <vxml:prompt>
        What vegetable toppings would you like?
      </vxml:prompt>
      <vxml:grammar src="vegtoppings.jsgf"/>
      <vxml:catch event="help nomatch noinput">
        For example, say mushrooms and peppers.
      </vxml:catch>
      <vxml:filled>
        <vxml:assign name="varVegetableToppings" expr="vtoppingsvegetable"/>
      </vxml:filled>
    </vxml:field>
  </vxml:form>
```

Quelle: IBM (2003b)

Abbildung 17: X+V – Grammatik Datei

```
vegtoppings.jsgf
grammar vegitable_toppings;
<veggies> = olives | mushrooms | onions | peppers;
public <toppings> =
    <NULL> {
            this.$value="";
    }
    (<veggies> [and]{
            this.$value = this.$value + " " + $veggies;
    })+;
```

Quelle: IBM (2003b)

Der Ausführungsteil der X+V Datei beinhaltet den Code der festlegt welche Anweisungen für welches *Event* ausgeführt werden. Dieser Teil beinhaltet auch den *Event Handler* (vgl. Abbildung 18)

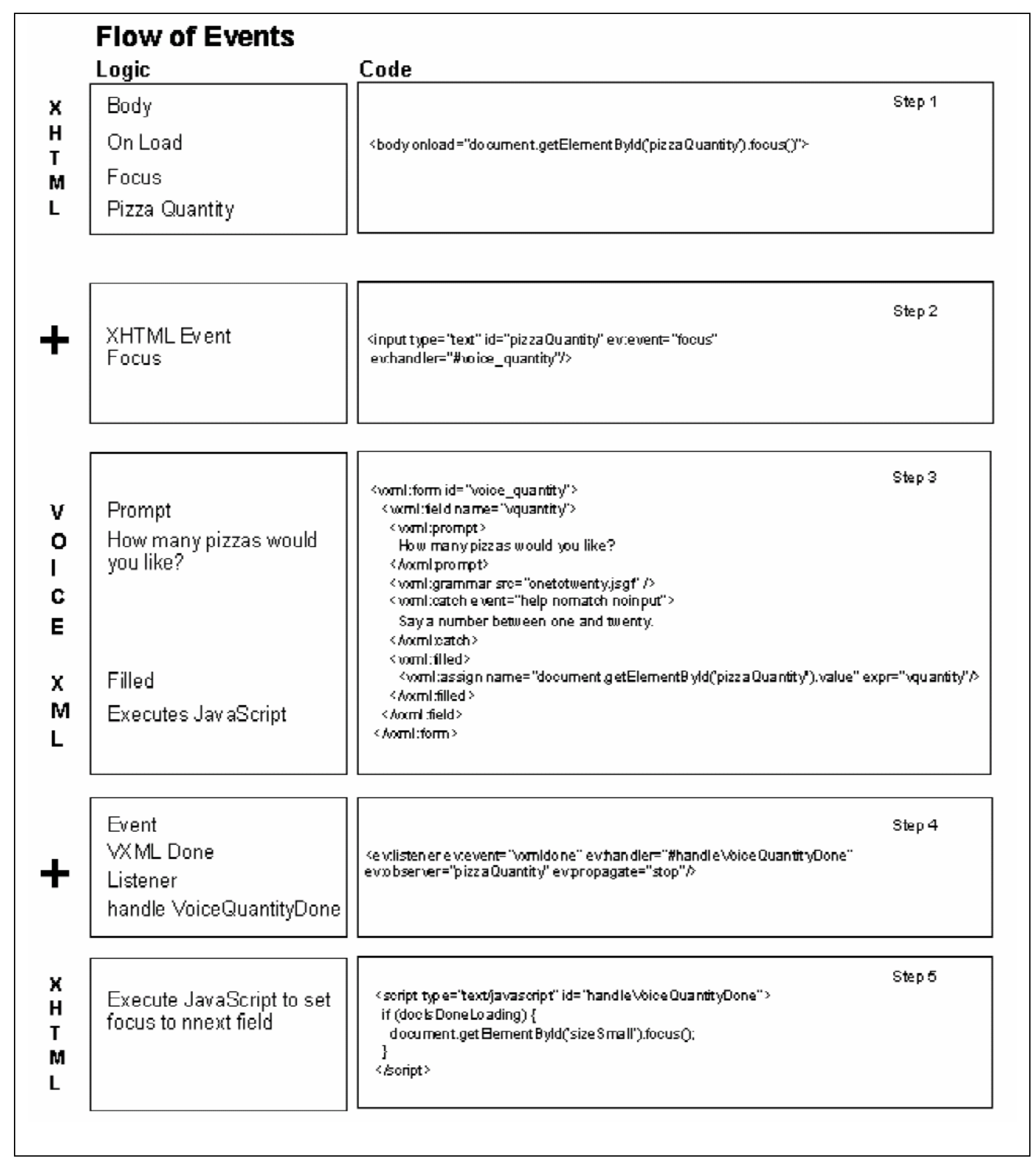
Abbildung 18: X+V – Ausführungsteil

```
<b>Quantity:</b><br/>
<input type="text" id="pizzaQuantity" ev:event="focus" ev:handler="#voice_quantity"/>
<ev:listener ev:event="vxmldone" ev:handler="#handleVoiceQuantityDone"
ev:observer="pizzaQuantity" ev:propagate="stop"/>
```

Quelle: IBM (2003b)

Die einzelnen Teile der XHTML Seite werden systematisch anhand des Event Handlers und der Ausführungsanweisungen abgearbeitet. Abbildung 19 zeigt einen Ereignisfluss anhand eines Eingabefeldes des XHTML Formulars.

Abbildung 19: X+V – Ereignis-Fluss



Quelle: IBM (2003b)

Der Vorteil der Trennung einer X+V Seite in einen visuellen Teil und einen Sprachteil ist, dass, wie oben bereits erwähnt, die Applikationen auch von Endgeräten genutzt werden können, die keine Sprachsteuerung unterstützen. Weiters ist es möglich den VoiceXML Teil in einer separaten Datei zu codieren. Somit wäre es möglich den visuellen und den Sprachteil unabhängig voneinander zu entwickeln. Dies wiederum ermöglicht die Wiederverwendung von einzelnen VoiceXML Teilen (vgl. IBM, 2004).

Derzeit sind zwei multimodale Browser die X+V unterstützen für Desktop PCs verfügbar. Es handelt sich dabei um *Opera 8* und *NetFront v3.1*, wobei beide im Bereich Voice Recognition und Text To Speech auf die Technologie der IBM zurückgreifen (vgl. Opera, 2005 und ACCESS, 2005). Beide Browser sind Freeware (Opera mit der Einschränkung eines Werbebanners) und von NetFront gibt es auch eine 45-Tage Testversion für das Betriebssystem Pocket PC 2003. Von Opera ist eine Version für *Sharp Zaurus* erhältlich (vgl. IBM, 2005). Weiters werden von kommerziellen Anbietern wie z.B. der Firma *TOP System* Browser angeboten die ebenfalls X+V unterstützen und den Vorteil haben, dass sie auch Applikationen in deutscher Sprache ermöglichen (siehe Kapitel 3).

3 Realisierung der Beispieltransaktion

3.1 Anforderungen des Auftraggebers

Ziel war es, eine Beispieltransaktion unter Verwendung des WAS 6.2 als Entwicklungsumgebung zu programmieren. Dazu werden einzelne Business Server Pages (BSPs), unter hauptsächlicher Verwendung von HTML, ABAP und JavaScript erstellt, wobei ABAP als serverseitige und JavaScript als clientseitige Scriptsprache eingesetzt wird. Um die Konzentration auf die eigentliche Entwicklungstätigkeit zu fokussieren wurde eine, aus fachlicher Sicht, einfache Transaktion gewählt. Die Transaktion soll zur Nullzählung im Zuge einer permanenten Inventur dienen. In weiterer Folge soll aus der vorhandenen Beispieltransaktion ein Template abgeleitet werden um die Entwicklung sprachgesteuerter Transaktionen in Zukunft zu erleichtern bzw. zu beschleunigen.

Die benötigten VoiceXML Befehle sollen manuell in die BSPs integriert werden. Diese Integration wird durch die Verwendung des X+V Standards möglich (siehe Subkapitel 2.4). Die BSPs sollen in einem späteren Produktivbetrieb vorrangig von dem *XVBrowser* der Firma TOP System unter Verwendung der mitgelieferten Sprach-Engine dargestellt werden. Um jedoch so nahe wie möglich am Standard zu bleiben und die Möglichkeiten der verfügbaren Freeware auszuloten, soll zuerst damit begonnen werden die Transaktion für den multimodalen Browser NetFront zu entwickeln und später davon eine Version für den kommerziellen Browser abzuleiten. Da es sich bei dem XVBrowser nicht um einen vollwertigen Browser sondern um einen abgespeckten Browser für das Betriebssystem Pocket PC 2003 handelt, muss geprüft werden in welchem Unfang X+V unterstützt wird.

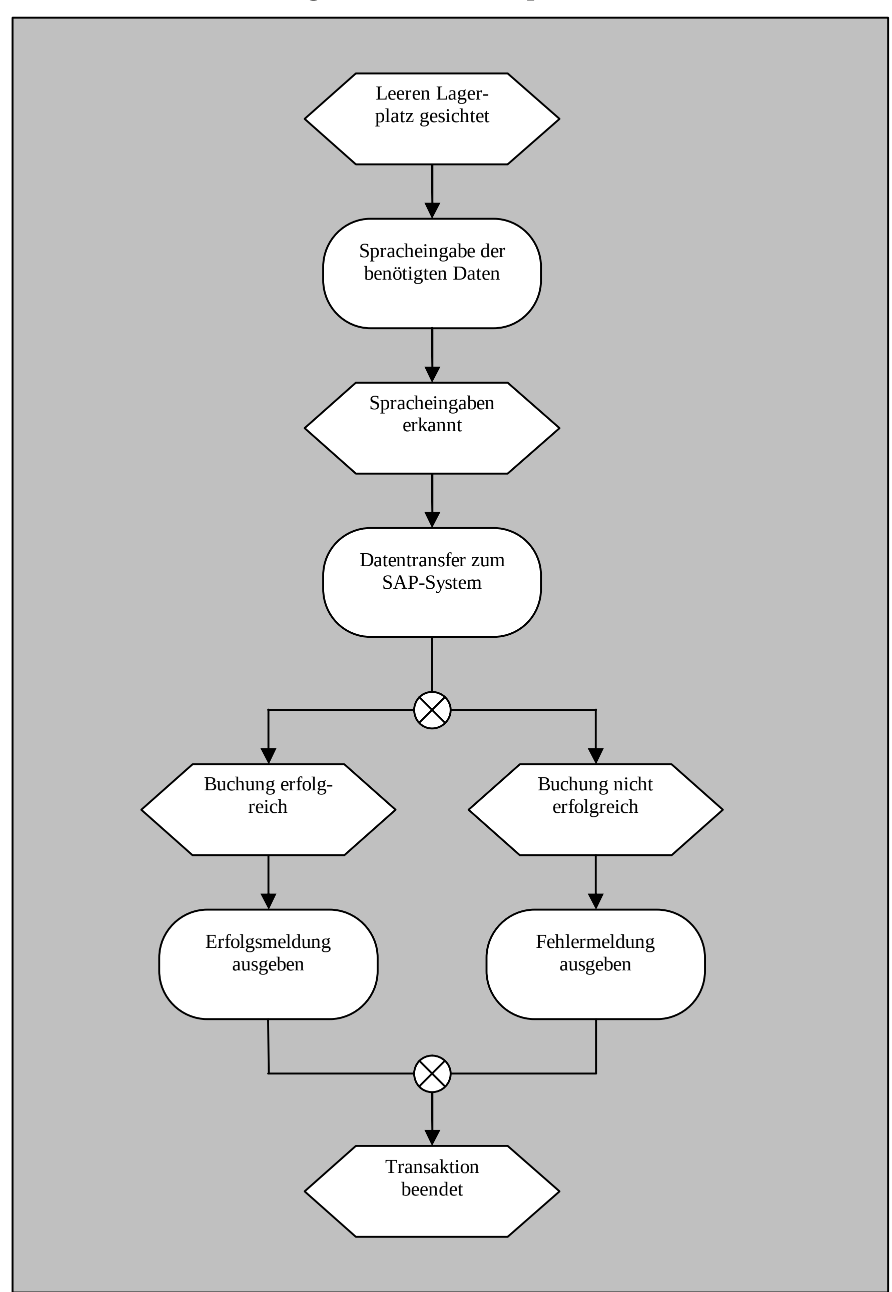
Als Handheld wurde ein HP iPAQ mit Head-Set von der Firma TOP System zur Verfügung gestellt, da dieses Gerät zurzeit vorrangig Verwendung in den realisierten Kundenprojekten findet.

3.2 Design der Beispieltransaktion

3.2.1 Transaktionsablauf

Der betriebswirtschaftliche Ablauf lässt sich aufgrund der bereits erwähnten Fokussierung auf den X+V Standard und die damit verbundene Einfachheit der Transaktion sehr leicht in einer ereignisgesteuerten Prozesskette darstellen (vgl. Abbildung 20).

Abbildung 20: EPK der Beispieltransaktion



Quelle: Eigene Abbildung

3.2.2 Sprachablauf NetFront

Tabelle 5: Sprachablauf NetFront

ID	Wer?	Sprachausgabe bzw. Spracheingabemöglichkeiten	GOTO
1		„Please enter Storage Type“	2
2		„<Zahl><Zahl><Zahl>“	3
		„Help“	8
3		„Confirm <Zahl><Zahl><Zahl>” „Please enter Storage Place”	4
4		„<Zahl>“	5
		„Help“	9
5		„Please confirm Place <Zahl>“	6
6		„Next“ um zu bestätigen und die Daten an SAP zu senden	7
		„Cancel“ um die Daten zu korrigieren	1
7		„Zero Count executed“	1
		„Errorcode <Zahl><Zahl><Zahl>“	2
8		„Allowed values are: 001, 002 or 003.“	2
9		„Allowed values are the numbers from 1 to 9.“	4

Quelle: Eigene Tabelle



.... Applikation

.... Anwender

Die Spracheingabe in obigem Ablauf erfolgt je Eingabefeld (d.h. sobald das jeweilige Eingabefeld aktiv ist und die entsprechende Meldung vom System abgesetzt wurde). Darüber hinaus kann der User auch bei jeder Eingabeaufforderung feldübergreifende

Befehle sprechen. In der vorliegenden Transaktion besteht die Möglichkeit mit „Back“ in das Hauptmenü zu verzweigen oder mit „Refresh“ die Transaktion neu zu starten.

3.2.3 *Sprachablauf XVBrowser*

Tabelle 6: Sprachablauf XVBrowser

ID	Wer?	Sprachausgabe bzw. Spracheingabemöglichkeiten	GOTO
1		„Bitte Lagertyp eingeben“	2
2		„Typ <Zahl><Zahl><Zahl>“	3
		„Wiederholen“	1
3		„Bitte Lagerplatz eingeben”	4
4		„Platz <Zahl>“	5
		„Wiederholen“	3
5		„Bestätige Platz <Zahl>“	6
6		„Wiederholen“	5
		„Weiter“ um zu bestätigen und die Daten an SAP zu senden	7
		„Abbrechen“ um die Daten zu korrigieren	1
7		„Nullzählung durchgeführt“	1
		„Fehlercode <Zahl><Zahl><Zahl>“	2

Quelle: Eigene Tabelle


.... Applikation

.... Anwender

Die Spracheingabe in obigem Ablauf erfolgt wie bei NetFront je Eingabefeld (d.h. sobald das jeweilige Eingabefeld aktiv ist und die entsprechende Meldung vom System abgesetzt wurde). Darüber hinaus kann der User auch in der vorliegenden Transaktion bei jeder Eingabeaufforderung feldübergreifende Befehle sprechen. Es besteht die Möglichkeit mit

„Zurück“ in das Hauptmenü zu verzweigen oder mit „Refresh“ die Transaktion neu zu starten.

3.3 Programmierung der Business Server Pages

3.3.1 Visueller Ablauf mit NetFront

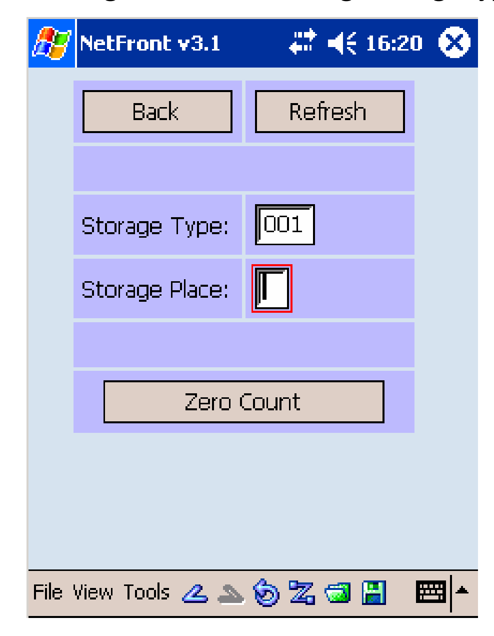
Beim Starten des Menüs (vgl. Abbildung 21) wird ein kurzer Begrüßungstext gesprochen und man wird aufgefordert eine Auswahl zu treffen. Durch die gesprochene Auswahl eines Buttons gelangt man auf das Einstiegsbild der jeweiligen Transaktion. Im vorliegenden Fall kann die Beispieltransaktion *Zero Count* oder die Variante *Voice only* ausgewählt werden. Die weiteren Buttons dienen zur Demonstration bzw. als Platzhalter für spätere Entwicklungen.

Abbildung 21: NetFront – Hauptmenü

Quelle: Eigene Abbildung

Im nächsten Bild (vgl. Abbildung 22) wird der User aufgefordert, den Lagertyp einzugeben. Optional kann der Hilfetext zum Eingabefeld aufgerufen werden. Nach der Eingabe des Lagertyps wird dieser bestätigt und der Cursor springt in das nächste Eingabefeld und die entsprechende Eingabeaufforderung wird ausgegeben.

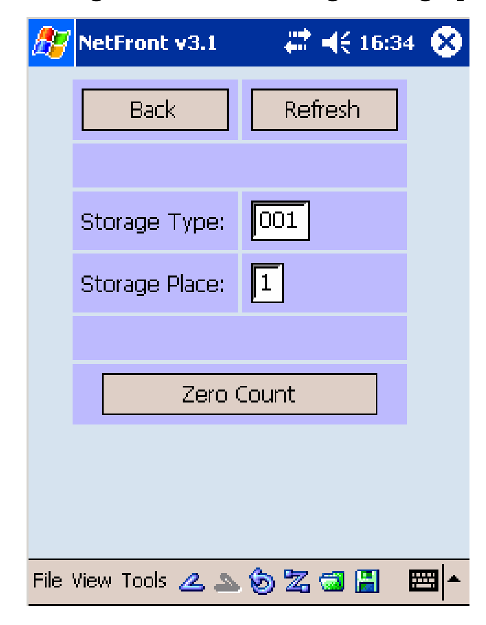
Abbildung 22: NetFront – Eingabe Lagertyp



Quelle: Eigene Abbildung

Der User wird anschließend aufgefordert, den Lagerplatz einzugeben (vgl. Abbildung 23). Optional kann der Hilfetext zum Eingabefeld aufgerufen werden. Nach der Eingabe des Lagerplatzes wird man aufgefordert den leeren Lagerplatz zu bestätigen. Nach erfolgter Bestätigung wird die Hintergrundverarbeitung angestoßen.

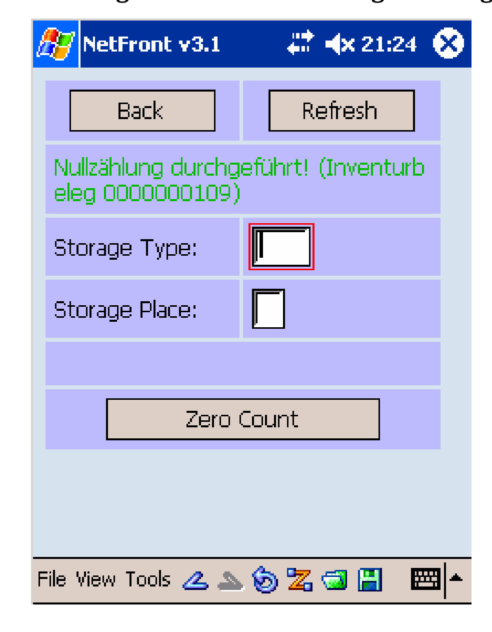
Abbildung 23: NetFront – Eingabe Lagerplatz



Quelle: Eigene Abbildung

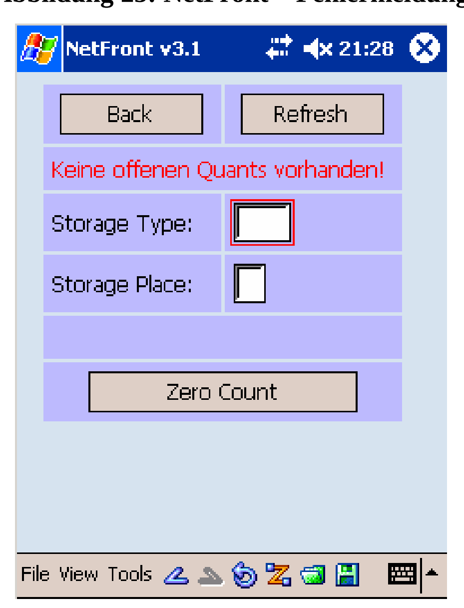
Nach erfolgreicher Buchung erfolgt die entsprechende Sprachausgabe und auch visuell wird die Meldung am Display ausgegeben (vgl. Abbildung 24). Im Fehlerfall wird ebenfalls eine entsprechende Meldung ausgegeben. (vgl. Abbildung 25).

Abbildung 24: NetFront – Erfolgsmeldung



Quelle: Eigene Abbildung

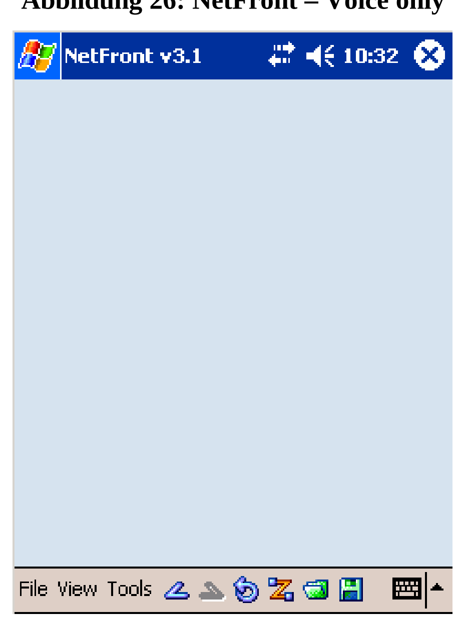
Abbildung 25: NetFront – Fehlermeldung



Quelle: Eigene Abbildung

Der Ablauf der Variante „Voice only“ ist identisch mit der Variante mit visueller Anzeige, außer dass eben eine leere Seite am Display des Browsers erscheint (vgl. Abbildung 26). Diese Variante wurde zu Demonstrationszwecken entwickelt um deutlich zu veranschaulichen, dass auch eine reine Sprachsteuerung möglich ist.

Abbildung 26: NetFront – Voice only



Quelle: Eigene Abbildung

3.3.2 Spezifischer Sourcecode für NetFront

Wie bereits erwähnt wird die Transaktion durch die Programmierung einer Business Server Page (BSP) realisiert. Dabei handelt es sich um eine XHTML Datei die auf dem Web Application Server (WAS) abgelegt ist. Begonnen wird mit dem standardmäßigen Aufbau des Grundgerüsts einer HTML Seite mit der entsprechenden Namespace Declaration.

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:vxml="http://www.w3.org/2001/vxml"
      xmlns:ev="http://www.w3.org/2001/xml-events">
<head>
      <title>Flexus Inventory Manager</title>
```

Die visuelle Komponente der Einstiegsseite mit dem Auswahlmenü wird über ein standardmäßiges HTML Formular im Body-Bereich realisiert.

```
<body bgcolor="#d4e2ee" leftmargin="5" topmargin="5"
onload="javascript:doOnLoad();">

<form name="form1">

     <table cellpadding="5" align="center">
          <tr bgcolor="#BBBBFF">
               <td align="center">
                    <img src="images/Flexus_logo_200.gif"/>
                    <input type="hidden" name="input" id="input"
                    ev:event="focus" ev:handler="#voiceInput"/></td>
          </tr>
          <tr bgcolor="#BBBBFF">
               <td align="center">
                    <input style="text-align:center; width:150" type="button"
                    name="zerocount" value="Zero Count"
                    onClick="self.location.href='inv_wm_zero_netfront.html'"/>
               </td>
          </tr>
          <tr bgcolor="#BBBBFF">
               <td align="center">
                    <input style="text-align:center; width:150" type="button"
                    name="transaction2" value="Voice only"

     onClick="self.location.href='inv_wm_zero_voiceonly.html'"/>
          </td>
          </tr>
          <tr bgcolor="#BBBBFF">
               <td align="center">
                    <input style="text-align:center; width:150" type="button"
                    name="transaction3" value="Transaction 3"
                    onClick="self.location.href='index.html'"/></td>
          </tr>
          <tr bgcolor="#BBBBFF">
               <td align="center">
                    <input style="text-align:center; width:150" type="button"
                    name="transaction4" value="Transaction 4"
                    onClick="self.location.href='index_netfront.html'"/></td>
          </tr>
          <tr bgcolor="#BBBBFF">
               <td align="center">
                    <input style="text-align:center; width:150" type="button"
                    name="transaction5" value="Transaction 5"
                    onClick="self.location.href='index_netfront.html'"/></td>
          </tr>
     </table>

</form>

</body>

</html>
```

Im Head-Bereich wird das JavaScript definiert, welches beim Laden der Seite aufgerufen wird. Es dient dazu den Fokus auf das erste Eingabefeld zu setzen und somit die Verarbeitung des VoiceXML Formulars zu starten.

```
<script type="text/javascript">
     var recognized="" ;
```

```
	function doOnLoad() {
		document.form1.input.focus() ;
	}

</script>
```

Weiters werden im Kopfbereich die einzelnen Eingabefelder mit deren Sprachausgaben in einem VoiceXML Formular definiert. Mit der (in diesem Fall eingebundenen) Grammatik wird festgelegt, welche Spracheingaben zulässig sind.

```
<vxml:form id="voiceInput">
	<vxml:field name="v_input">
		<vxml:prompt>
			Welcome to the Inventory Manager, powered by Flexus!
			Please choose a transaction.
		</vxml:prompt>
		<vxml:grammar>
			<![CDATA[
				#JSGF V1.0;
				grammar browser;
				public <browser> = Zero Count | Voice only | Stop;
			]]>
		</vxml:grammar>
		<vxml:filled>
			<vxml:assign name="recognized" expr="v_input"/>
			confirm <vxml:value expr="v_input"/>
		</vxml:filled>
	</vxml:field>
</vxml:form>
```

Die Spracheingaben werden vorerst in einer Variablen gesichert und können anschließend über ein JavaScript überprüft und eventuell in das Eingabefeld des HTML-Formulars geschoben werden. Im vorliegenden Fall wird nur die Folgeverarbeitung angestoßen.

```
<script type="text/javascript" id="scriptInput">
	if(recognized=="Zero Count")
		document.form1.zerocount.click();
	else if (recognized=="Voice only")
		document.form1.voiceonly.click();
	else{
	}
</script>
```

Damit das Script auch tatsächlich verarbeit wird, wird das dafür nötige Event im Event Listener für das betroffene (in diesem Fall verborgene) Eingabefeld des HTML Formulars festgelegt.

```
<ev:listener ev:event="vxmldone" ev:handler="#scriptInput"
ev:observer="input" ev:propagate="stop" />
```

Die visuelle Komponente der Beispieltransaktion wird in einer weiteren X+V Seite (mit derselben Namespace Declaration) ebenfalls über ein standardmäßiges HTML Formular im

Body-Bereich realisiert. Zu Beachten ist, dass der Button zum Absenden des Formulars einen Namen nach WAS Konvention von der Form *onInputProcessing(<event_id>)* hat. Die ABAP Codeabschnitte für serverseitiges Scripting sind blau dargestellt. Da nach der Verbuchung die gleiche Seite (für etwaige weitere Lagerplätze) nochmals aufgerufen wird, wird geprüft, ob eine entsprechende Meldung visuell ausgegeben werden muss.

```
<body bgcolor="#d4e2ee" leftmargin="5" topmargin="5"
onload="javascript:doOnLoad();">

<form name="form1" method="post" onSubmit="return checkFormular();">

     <table cellpadding="5" align="center">
          <tr bgcolor="#BBBBFF">
               <td align="center">
                    <input style="text-align:center; width:80" type="button"
                    name="Back" value="Back"
                    onClick="self.location.href='index_netfront.html'"/>
               </td>
               <td align="center">
                    <input style="text-align:center; width:80" type="button"
                    name="Refr" value="Refresh"
                    onClick="self.location.href='inv_wm_zero_netfront.html'"/>
               </td>
          </tr>
          <tr bgcolor="#BBBBFF">
               <td colspan="2">
                    <% if not ivnum is initial. %>
                         <font color="#00BB00"><%= return-message %></font>
                    <% elseif subrc ne 0. %>
                         <font color="#FF0000"><%= return-message %></font>
                    <% elseif subrc eq 0. %>
                         <br/>
                    <% endif. %>
               </td>
          </tr>     <tr bgcolor="#BBBBFF">
               <td>Storage Type:</td>
               <td>
                    <input type="text" name="lgtyp" maxlength="3" size="3"
                    id="lgtyp" ev:event="focus"
ev:handler="#voiceLgtyp"/></td>
          </tr>
          <tr bgcolor="#BBBBFF">
               <td>Storage Place:</td>
               <td>
                    <input type="text" name="lgpla" maxlength="1" size="1"
                    id="lgpla" ev:event="focus"
ev:handler="#voiceLgpla"/></td>
          </tr>
          <tr bgcolor="#BBBBFF">
               <td colspan="2"><br/>
                    <input type="hidden" name="confirm" id="confirm"
                    ev:event="focus" ev:handler="#voiceConfirm"/></td>
          </tr>
          <tr bgcolor="#BBBBFF">
               <td colspan="2" align="center">
                    <input style="text-align:center; width:150" type="submit"
                    name="onInputProcessing(BOOK)" id="Book"
                    value="Zero Count"/></td>
          </tr>
     </table>
```

```
</form>

</body>

</html>
```

Für die Variante „Voice only“ müssen für einen korrekten Ablauf der Seite bzw. des VoiceXML Formulars ebenfalls alle Elemente des HTML Formulars definiert werden, allerdings werden sie unsichtbar gemacht.

```
<body bgcolor="#d4e2ee" leftmargin="5" topmargin="5"
onload="javascript:doOnLoad();">

<form name="form1" method="post">

     <input style="visibility:hidden" type="button" name="Back" value="Back"
     onClick="self.location.href='index_netfront.html'"/></td>
     <input style="visibility:hidden" type="button" name="Refr" value="Refresh"
     onClick="self.location.href='inv_wm_zero_voiceonly.html'"/></td>

     <input type="hidden" name="lgtyp" maxlength="3" size="3" id="lgtyp"
     ev:event="focus" ev:handler="#voiceLgtyp"/></td>
     <input type="hidden" name="lgpla" maxlength="1" size="1" id="lgpla"
     ev:event="focus" ev:handler="#voiceLgpla"/></td>
     <input type="hidden" name="confirm" id="confirm"
     ev:event="focus" ev:handler="#voiceConfirm"/></td>
     <input style="visibility:hidden" type="submit"
     name="onInputProcessing(BOOK)" id="Book" value="Zero Count"/></td>

</form>

</body>

</html>
```

Die einzelnen Eingabefelder mit Sprachausgabe und Spracherkennung werden im Kopfbereich der X+V Seite definiert. Die erlaubten Spracheingaben werden zur besseren Übersicht wieder direkt in den XHTML Code eingebunden. Auch hier wird ABAP für serverseitiges Scripting verwendet, um festzustellen, ob die Seite nach einer erfolgreichen Verbuchung (also nicht zum ersten Mal) aufgerufen wird und eine Meldung ausgegeben werden soll.

```
<vxml:form id="voiceLgtyp">
     <vxml:field name="v_lgtyp">
          <vxml:prompt>
               <% if not ivnum is initial.%>
                    Zero count executed!
                    Please enter storage type
               <% elseif subrc ne 0.%>
                    error code <%= return-number %>
               <% elseif subrc eq 0.%>
                    Please enter storage type
               <% endif.%>
          </vxml:prompt>
```

```
          <vxml:grammar>
               <![CDATA[
                    #JSGF V1.0;
                    grammar digits;
                    public <digits> = 001 | 002 | 003 | back | refresh;
               ]]>
          </vxml:grammar>
          <vxml:catch event="help">
               Allowed Values are: 001, 002 and 003.
          </vxml:catch>
          <vxml:filled>
               <vxml:assign name="recognized" expr="v_lgtyp"/>
               confirm <vxml:value expr="v_lgtyp"/>
          </vxml:filled>
     </vxml:field>
</vxml:form>

<vxml:form id="voiceLgpla">
     <vxml:field name="v_lgpla">
          <vxml:prompt>Please enter Storage Place</vxml:prompt>
          <vxml:grammar>
               <![CDATA[
                    #JSGF V1.0;
                         grammar digits;
                         public <digits> = 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
                         back | refresh;
               ]]>
          </vxml:grammar>
          <vxml:catch event=" help">
               Allowed Values are: 1, 2, 3, 4 and 5.
          </vxml:catch>
          <vxml:filled>
               <vxml:assign name="recognized" expr="v_lgpla"/>
          </vxml:filled>
     </vxml:field>
</vxml:form>

<vxml:form id="voiceConfirm">
     <vxml:field name="v_confirm">
          <vxml:prompt>
               <vxml:value expr="getPromptText(document.form1.lgpla.value)"/>
          </vxml:prompt>
          <vxml:grammar>
               <![CDATA[
                    #JSGF V1.0;
                    grammar confirm;
                    public <confirm> = next | cancel;
               ]]>
          </vxml:grammar>
          <vxml:catch event="nomatch help">
                    If the data is correct say "NEXT". If you want to change
                    something, say "CANCEL".
          </vxml:catch>
          <vxml:filled>
               <vxml:assign name="recognized" expr="v_confirm"/>
          </vxml:filled>
     </vxml:field>
</vxml:form>
```

Die Übergaben der Spracheingabe in die korrespondierenden Formularfelder sowie die *Aktivierungen* der nächsten Eingabefelder werden mittels JavaScripts durchgeführt. Diese werden ebenfalls im Kopfteil der X+V Seite definiert.

```
<script type="text/javascript" id="scriptLgtyp">
	if(recognized!="") {
		if(verifyLgtyp(recognized)) {
			document.form1.lgtyp.value=recognized ;
			document.form1.lgpla.focus() ;
		}
	}
</script>

<script type="text/javascript" id="scriptLgpla">
	if(recognized!="") {
		if(verifyLgpla(recognized)) {
			document.form1.lgpla.value=recognized ;
			document.form1.confirm.focus() ;
		}
	}
</script>

<script type="text/javascript" id="scriptConfirm">
	if(recognized!="") {
		if(recognized=="next")
			doOnSubmit() ;
		else if(recognized=="cancel")
			document.form1.Refr.click() ;
	}
</script>
```

Im Kopf der HTML-Seite werden auch noch allgemeine Scriptfunktionen, wie z.B. das Handling der feldübergreifenden Befehle, definiert. Weiters wird pro Eingabefeld ein Script codiert um eventuell notwendige Prüfungen oder Berechnungen durchzuführen. Zu beachten ist die Funktion *checkFormular*, welche überprüft ob auch tatsächlich alle obligatorischen Felder gefüllt wurden. Diese Funktion wird sowohl bei Drücken des *Absende*-Buttons des Formulars als auch am Ende der Sprachsteuerung aufgerufen.

```
<script type="text/javascript">
	var recognized="" ;

	function checkButtons(ok_code){
		if (ok_code=="back"){
			document.form1.Back.click();
			return false;
		}else if (ok_code=="refresh"){
			document.form1.Refr.click();
			return false;
		}else
			return true;
	}

	function verifyLgtyp(recognized) {
		if (checkButtons(recognized))
			return true;
		else
			return false;
```

```
}

function verifyLgpla(recognized) {
     if (checkButtons(recognized))
          return true;
     else
          return false;
}

function getPromptText(place) {
     return "Please Confirm empty place "+place ;
}

function doOnLoad() {
     document.form1.lgtyp.focus() ;
}

function doOnSubmit() {
     if (checkFormular())
          document.getElementById('Book').click();
}

function checkFormular() {
     if (!document.form1.lgtyp.value){
          document.form1.lgtyp.focus();
          return false;
     }else if (!document.form1.lgpla.value){
          document.form1.lgpla.focus();
          return false;
     }else{
          return true;
     }
}</script>
```

Damit diese Scripts jeweils zum richtigen Zeitpunkt aufgerufen werden, wird das *Document Object Model (DOM)* genutzt und der entsprechende Event Listener im Kopf-Bereich codiert.

```
<ev:listener ev:event="vxmldone" ev:handler="#scriptLgtyp"
ev:observer="lgtyp" ev:propagate="stop" />

<ev:listener ev:event="vxmldone" ev:handler="#scriptLgpla"
ev:observer="lgpla" ev:propagate="stop" />

<ev:listener ev:event="vxmldone" ev:handler="#scriptConfirm"
ev:observer="confirm" ev:propagate="stop" />
```

Die Business Logik wird im Event Handler des Web Application Servers in ABAP codiert. Dabei wird im Normalfall über einen *Remote Function Call (RFC)* im angekoppelten Backend-System die Aktualisierung und Verbuchung der SAP Standard Belege zur Inventur angestoßen. In der vorliegenden Beispieltransaktion wird dies durch einen Kommentar demonstriert, die tatsächliche Codierung erfolgt in der Beispieltransaktion für den XVBrowser (siehe Subkapitel 3.3.4).

```
case event_id.
```

```
     when 'BOOK'.
* An dieser Stelle wird im Normalfall ein RFC ausgeführt!
endcase.
```

3.3.3 Visueller Ablauf mit dem XVBrowser

Das Menü für den XVBrowser ist, was die Funktionalität betrifft, sehr ähnlich dem NetFront Menü (vgl. Abbildung 27). Es wird ebenfalls ein kurzer Begrüßungstext gesprochen und man wird aufgefordert eine Auswahl zu treffen. Durch die gesprochene Auswahl eines Buttons gelangt man auf das Einstiegsbild der jeweiligen Transaktion. Im vorliegenden Fall kann nur die Beispieltransaktion *Nullzählung* ausgewählt werden. Die weiteren Buttons dienen wie bei der NetFront Applikation zur Demonstration bzw. als Platzhalter für spätere Entwicklungen.

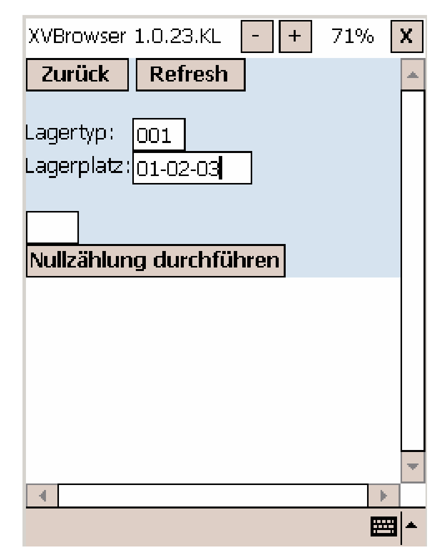
Abbildung 27: XVBrowser – Startmenü

XVBrowser 1.0.23.KL - + 31% X
Nullzählung
Funktion 2
Funktion 3
Funktion 4
Funktion 5

Quelle: Eigene Abbildung

Da der XVBrowser unsichtbare Felder nicht optimal unterstützt, wird das *Bestätigungsfeld*, also das letzte Feld für die Sprachaus- bzw. Spracheingabe, immer angezeigt. In der Maske der Transaktion *Nullzählung* werden dann der Lagertyp und der Lagerplatz eingegeben (vgl. Abbildung 28).

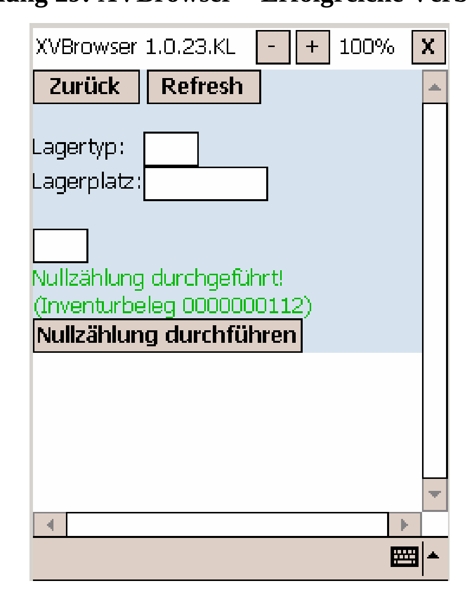
Abbildung 28: XVBrowser – Dateneingabe



Quelle: Eigene Abbildung

Nachdem man die Eingaben bestätigt hat, wird die Hintergrundverarbeitung, also die Verbuchung im SAP System angestoßen und die Transaktion für weitere Lagerplätze neu gestartet. Bei erfolgreicher Aktualisierung des entsprechenden Inventurbeleges, wird eine akustische Meldung ausgeben und am Display die Nummer des Inventurbeleges angezeigt (vgl. Abbildung 29). Sollte ein Fehler aufgetreten sein wird akustisch ein Fehlercode und am Display eine Erklärung ausgegeben (vgl. Abbildung 30).

Abbildung 29: XVBrowser – Erfolgreiche Verbuchung



Quelle: Eigene Abbildung

Abbildung 30: XVBrowser - Unerwarteter Fehler

Quelle: Eigene Abbildung

3.3.4 *Spezifischer Sourcecode für den XVBrowser*

Wie bei Netfront wird die Menüseite als BSP am WAS mit dem Grundgerüst einer normalen HTML-Seite abgelegt und die entsprechende Namespace Declaration für X+V Seiten codiert.

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:vxml="http://www.w3.org/2001/vxml"
      xmlns:ev="http://www.w3.org/2001/xml-events">
<head>
      <title>Flexus Inventory Manager</title>
```

Beim Codieren des visuellen Formulars besteht der einzige Unterschied in der Deklaration des einzigen Eingabefelds. Da der XVBrowser das Fokussieren unsichtbarer Felder nicht unterstützt muss hier ein sichtbares Feld codiert werden. Auch die Möglichkeiten der visuellen Gestaltung sind bei Verwendung des XVBrowsers sehr eingeschränkt, daher wird auf die Nutzung einer (unsichtbaren) Tabelle verzichtet.

```
<body bgcolor="#D4E2EE" onload="javascript:doOnLoad();">

<form name="form1" method="post">

      <input type="button" name="zero" value="Nullzählung"
      onClick="document.location.href='inv_wm_zero_topsystem.html'"/><br/>

      <input type="button" name="tra2" value="Funktion 2"
      onClick="document.location.href='index_topsystem.html'"/><br/>

      <input type="button" name="tra3" value="Funktion 3"
```

```
    onClick="document.location.href='index_topsystem.html'"/><br/>

    <input type="button" name="tra4" value="Funktion 4"
    onClick="document.location.href='index_topsystem.html'"/><br/>

    <input type="button" name="tra5" value="Funktion 5"
    onClick="document.location.href='index_topsystem.html'"/><br/>

    <br/>
    <input type="text" name="confirm" id="confirm"
    ev:event="focus" ev:handler="#voiceConfirm"/>

</form>

</body>

</html>
```

Beim Laden der Seite wird ein JavaScript ausgeführt um den Fokus auf das Eingabefeld zu legen. Dieses Script wird im Kopfbereich der HTML-Seite codiert.

```
<script type="text/javascript">
    var recognized="" ;

    function doOnLoad() {
        document.form1.confirm.focus() ;
    }

</script>
```

Das VoiceXML Formular wird ebenfalls im Kopfbereich definiert. Weiters wird in JavaScript ein Ablauf zur Prüfung und Verarbeitung der Spracheingaben sowie der benötigte Event Listener codiert. Je nach Befehl wird der entsprechende Button gedrückt, um zur nächsten BSP zu gelangen. In der vorliegenden BSP gibt es nur eine Auswahlmöglichkeit.

```
<vxml:form id="voiceConfirm">
    <vxml:field name="v_confirm">
        <vxml:prompt>
            bitte funktion waehlen
        </vxml:prompt>
            <vxml:grammar>
                <![CDATA[
                    #JSGF V1.0;
                    grammar confirm;
                    public <confirm> = nullzaehlung | wiederholen;
                ]]>
        </vxml:grammar>
        <vxml:filled>
            <vxml:assign name="recognized" expr="v_confirm"/>
        </vxml:filled>
    </vxml:field>
</vxml:form>

<script type="text/javascript" id="scriptConfirm">
    if(recognized!="") {
```

```
          if(recognized=="nullzaehlung")
               document.form1.zero.click() ;
     }
</script>

<ev:listener ev:event="vxmldone" ev:handler="#scriptConfirm"
ev:observer="confirm" ev:propagate="stop" />

</head>
```

Die Folgeseite wird ebenfalls als BSP mit derselben Namespace Declaration am WAS angelegt. Der ABAP Code (relevant bei mehrmaligem Aufruf der Seite bei Verbuchung mehrerer Lagerplätze) für serverseitiges Scripting ist wieder in blau dargestellt. Der hauptsächliche Unterschied zu Netfront ist die Codierung des Buttons zum Absenden des Formulars. Es wird ein normaler Button verwendet und die URL wird dynamisch mit allen benötigten Übergabewerten generiert. Dieses Vorgehen ist notwendig, da die Codierung der NetFront Applikation, wegen unzureichender Unterstützung von JavaScript, mit dem XVBrowser nicht funktioniert.

```
<body bgcolor="#D4E2EE" onload="javascript:doOnLoad();">

<form name="form1" method="post">

     <input type="button" name="Back" value="Zurück"
     onClick="document.location.href='index_topsystem.html'"/>

     <input type="button" name="Refr" value="Refresh"
     onClick="document.location.href='inv_wm_zero_topsystem.html'"/>
     <br/><br/>

     <table>
          <tr>
               <td> Lagertyp:</td>
               <td><input type="text" name="lgtyp" maxlength="3" size="3"
                    id="lgtyp" ev:event="focus"
ev:handler="#voiceLgtyp"/></td>
          </tr>
          <tr>
               <td>Lagerplatz:</td>
               <td><input type="text" name="lgpla" maxlength="8" size="8"
                    id="lgpla" ev:event="focus"
ev:handler="#voiceLgpla"/></td>
          </tr>
     </table>

     <br/>
     <input type="text" name="confirm" id="confirm" ev:event="focus"
     ev:handler="#voiceConfirm"/><br/>

     <% if not ivnum is initial. %>
          <font color="#00BB00"><%= return-message %></font><br/>
     <% elseif subrc ne 0. %>
          <font color="#FF0000"><%= return-message %></font><br/>
     <% endif. %>

     <input type="button" name="Book" value="Nullzählung durchführen"
     onClick="if (checkFormular()) {
     document.location.href='inv_wm_zero_topsystem.html?onInputProcessing(BOOK)
```

```
     &lgtyp='+document.form1.lgtyp.value+'&lgpla='+document.form1.lgpla.value;}
"
     />

</form>

</body>

</html>
```

Die Eingabefelder mit Sprachausgabe und Spracherkennung, inklusive der zugehörigen Scripts zum Anstoßen etwaiger Prüfungen und der Folgeverarbeitung, werden ebenfalls im Kopf der HTML Seite definiert. Wieder wird ABAP genutzt, um (im Falle eines mehrmaligen Aufrufs der Seite) eine Erfolgs- oder Fehlermeldung auszugeben.

Bei der Sprachausgabe ist zu erwähnen, dass der XVBrowser für die Ausgabe aufgezeichnete Audiodateien (Samples) verwendet. Daher wird im Text durch das Wort „pause“ eine Datei ohne Akustik abgespielt und es entsteht die gewünschte Verzögerung. Der Vorteil dieser Vorgehensweise ist, dass die Ausgabe besser klingt als das Vorlesen durch eine synthetische Stimme. Der Nachteil ist, dass für jede einzelne Sprachausgabe eine Datei aufgenommen werden muss. Diese Tatsache macht die Ausgabe der, standardmäßig in hoher Zahl auftretenden, SAP Systemmeldungen unmöglich. Das Problem wird durch die akustische Ausgabe eines entsprechenden Fehlercodes umgangen. Zur weiteren Information des Benutzers wird der tatsächliche Meldungstext ohnehin visuell ausgegeben.

Im Fall der Spracheingabe weist der XVBrowser ein paar spezielle Kriterien auf. An den normalen Text können durch das Anhängen eines Characterstrings in der Form *xxx* spezielle Befehle an die Sprach Engine mitgegeben werden. Im vorliegenden Fall wird durch *$S:-wort$* bewirkt dass nur Phrasen mit *wort* beginnend akzeptiert werden. Das Minus davor weist die Sprach Engine an, *wort* allerdings nicht an das VoiceXML Formular weiterzugeben. Ursprünglich für den Einsatz mit der SAP WEBConsole vorgesehen, können diese Extensions auch für den WAS verwendet werden (vgl. Topsystem, 2005). Die Grammatik für Befehle, die Formular übergreifend Geltung besitzen sollen (*zurück* und *refresh*), wird pro Eingabefeld angegeben. *Wiederholen* muss nicht angegeben werden, da diese Funktion bereits direkt in den XVBrowser implementiert ist.

```
<vxml:form id="voiceLgtyp">
     <vxml:field name="v_lgtyp">
          <vxml:prompt>
               <% if not ivnum is initial. %>
                    nullzaehlung durchgefuehrt pause
                    bitte lagertyp eingeben$S:-typ$
               <% elseif subrc ne 0. %>
                    fehlercode <%= return-number %> pause
               <% elseif subrc eq 0. %>
                    bitte lagertyp eingeben$S:-typ$
               <% endif. %>
          </vxml:prompt>
          <vxml:grammar>
               <![CDATA[
                    #JSGF V1.0;
                    grammar gram1;
```

```
                    public <gram1> = zurueck | refresh;
               ]]>
          </vxml:grammar>
          <vxml:filled>
               <vxml:assign name="recognized" expr="v_lgtyp"/>
          </vxml:filled>
     </vxml:field>
</vxml:form>

<script type="text/javascript" id="scriptLgtyp">
     if(recognized!="") {
          if(verifyLgtyp(recognized)) {
               document.form1.lgtyp.value=recognized ;
               document.form1.lgpla.focus() ;
          }
     }
</script>


<vxml:form id="voiceLgpla">
     <vxml:field name="v_lgpla">
          <vxml:prompt>
               bitte lagerplatz eingeben$S:-platz$
          </vxml:prompt>
          <vxml:grammar>
               <![CDATA[
                    #JSGF V1.0;
                    grammar gram1;
                    public <gram1> = zurueck | refresh;
               ]]>
          </vxml:grammar>
          <vxml:filled>
               <vxml:assign name="recognized" expr="v_lgpla"/>
          </vxml:filled>
     </vxml:field>
</vxml:form>

<script type="text/javascript" id="scriptLgpla">
     if(recognized!="") {
          if(verifyLgpla(recognized)) {
               document.form1.lgpla.value=recognized ;
               document.form1.confirm.focus() ;
          }
     }
</script>


<vxml:form id="voiceConfirm">
     <vxml:field name="v_confirm">
          <vxml:prompt>
               <vxml:value expr="getPromptText(document.form1.lgpla.value)"/>
          </vxml:prompt>
          <vxml:grammar>
               <![CDATA[
                    #JSGF V1.0;
                    grammar confirm;
                    public <confirm> = weiter | abbrechen;
               ]]>
          </vxml:grammar>
          <vxml:filled>
               <vxml:assign name="recognized" expr="v_confirm"/>
          </vxml:filled>
     </vxml:field>
```

```
</vxml:form>

<script type="text/javascript" id="scriptConfirm">
     if(recognized!="") {
          if(recognized=="weiter")
               doOnSubmit() ;
          else if(recognized=="abbrechen")
               document.form1.Refr.click() ;
     }
</script>
```

Wie bei der Lösung mit Netfront werden im Kopfbereich noch die Scripts zur Feldprüfung sowie allgemeine Scripts definiert. Als Abschluss werden die Event Listener codiert.

```
<script type="text/javascript">
     var recognized="" ;

     function checkButtons(ok_code){
          if (ok_code=="zurueck"){
               document.form1.Back.click();
               return false;
          }else if (ok_code=="refresh"){
               document.form1.Refr.click();
               return false;
          }else
               return true;
     }

     function verifyLgtyp(recognized) {
          if (checkButtons(recognized))
               return true;
          else
               return false;
     }

     function verifyLgpla(recognized) {
          if (checkButtons(recognized))
               return true;
          else
               return false;
     }

     function getPromptText(place) {
          return "bestaetige platz "+place;
     }

     function doOnLoad() {
          document.form1.lgtyp.focus();
     }

     function doOnSubmit() {
          document.form1.Book.click();
     }

     function checkFormular() {
          if (!document.form1.lgtyp.value){
               document.form1.lgtyp.focus();
               return false;
          }else if (!document.form1.lgpla.value){
               document.form1.lgpla.focus();
```

```
                return false;
            }else{
                return true;
            }
        }

</script>


<ev:listener ev:event="vxmldone" ev:handler="#scriptLgtyp"
ev:observer="lgtyp"   ev:propagate="stop" />

<ev:listener ev:event="vxmldone" ev:handler="#scriptLgpla"
ev:observer="lgpla"   ev:propagate="stop" />

<ev:listener ev:event="vxmldone" ev:handler="#scriptConfirm"
ev:observer="confirm" ev:propagate="stop" />
```

Im Event Handler des Web Application Servers läuft zum Ereignis *OnInputProcessing* die Business-Logik, in ABAP codiert, ab. Dabei wird über einen Remote Function Call (RFC) im angekoppelten Backend-System die Aktualisierung und Verbuchung der SAP Standard Belege zur Inventur angestoßen.

```
case event_id.

    when 'BOOK'.

        clear: return, subrc, ivnum.

        CALL FUNCTION 'Z_SCF_FLEXUS_INV_NULL_GESAMT' destination 'IDE'
            EXPORTING
                iv_lgtyp       = lgtyp
                iv_lgpla       = lgpla
            IMPORTING
                ES_BAPIRET2    = return
                EV_SUBRC       = subrc
                EV_IVNUM       = ivnum.

        clear: lgtyp, lgpla.
endcase.
```

Durch den Funktionsbaustein *Z_SCF_FLEXUS_INV_NULL_GESAMT* werden alle nötigen Schritte, gekapselt und hintereinander, durch Aufruf der entsprechenden Funktionen abgearbeitet. Diese Funktionen wurden bereits, durch einzelne Unterprogramme (Formroutinen) oder weitere Funktionsbausteine, im Laufe vergangener Projekte durch die Entwickler der Flexus AG erstellt und werden im nachfolgenden Sourcecode nicht weiter einzeln erklärt.

Der Funktionsbaustein prüft zuerst ob für die Kombination Lagertyp und Lagerplatz bereits ein Inventurbeleg im System existiert. Existiert ein Beleg muss dieser gegebenenfalls noch aktiviert werden. Existiert kein Beleg wird über einen Funktionsbaustein einer angelegt und gleichzeitig aktiviert. Zur Durchführung der Zählung wird der gefundene oder angelegte Beleg (technisch) zur Bearbeitung durch einen anderen Benutzer gesperrt und die Daten

zum betreffenden Lagerplatz werden aus der Datenbank geholt. Für alle noch offenen Quants (kleinste Lagereinheit in SAP) wird die Nullmenge gebucht. Wurde der Platz bereits vollständig gezählt, wird das im Logbuch vermerkt und mit einer entsprechenden Meldung belegt. Erst nach erfolgter manueller Nachzählung kann in diesem Fall ein neuer Inventurbeleg angelegt und somit die Transaktion wieder verwendet werden.

```
FUNCTION z_scf_flexus_inv_null_gesamt.
*"----------------------------------------------------------------------
*"*"Lokale Schnittstelle:
*"  IMPORTING
*"     VALUE(IV_LGTYP) TYPE  LGTYP
*"     VALUE(IV_LGPLA) TYPE  LGPLA
*"  EXPORTING
*"     VALUE(ES_BAPIRET2) TYPE  BAPIRET2 (Letzte aufgezeichnete Meldung)
*"     VALUE(EV_SUBRC) TYPE  SYSUBRC     (Returncode)
*"     VALUE(EV_IVNUM) TYPE  LVS_IVNUM   (Nummer des Inventurbelegs)
*"  TABLES
*"      ET_LOGBUCH STRUCTURE  BAPIRET2 OPTIONAL
*"----------------------------------------------------------------------
*& Autor............: Bernhard Bugelmüller                              *
*& Erstellt am......: 07.09.2005                                        *
*& Geändert am......:                                                   *
*& Information......: WM-Inventur: Nullzählung                          *
*&                    Wird über sprachgesteuerte WAS-Seite aufgerufen!  *
*&                                                                      *
*& War der Aufruf des Funktionsbausteins erfolgreich, wird ev_subrc     *
*& auf 0 gesetzt, andernfalls wird ev_subrc auf 4 gesetzt.              *
*& Im Fehlerfall steht in es_bapiret2 die entsprechende Fehlermeldung.  *
*& Während des Ablaufs des Funktionsbaustein werden alle auftretenden   *
*& Meldungen in der Tabelle et_logbuch festgehalten.                    *
*"----------------------------------------------------------------------

* Datendeklarationen
  DATA: gv_subrc TYPE sysubrc,
        gv_lines TYPE i,
        gv_ivnum TYPE lvs_ivnum,
        gv_ivpos TYPE lvs_ivpos.

  DATA: lt_linv   TYPE TABLE OF linv,
        ls_linv   LIKE LINE OF lt_linv.


* Werte eingegeben?
  IF iv_lgtyp IS INITIAL or iv_lgpla IS INITIAL.
    ev_subrc = 4.
    gs_msg-msgty = 'E'.
    gs_msg-msgno = '075'.
    get_bapiret2 'DE'.
    APPEND es_bapiret2 TO et_logbuch.
    EXIT.
  ENDIF.

* Lagernummer zum User ermitteln
  gs_zbf_params-prog  = sy-repid.
  gs_zbf_params-uname = sy-uname.
  gs_zbf_params-spras = sy-langu.

  PERFORM zbf_get_params TABLES gt_zbf_params
                         USING  space gs_zbf_params.
```

```
* Logbucheintrag
  IF NOT gv_lgnum IS INITIAL.
    gs_msg-msgty = 'I'.
    gs_msg-msgno = '059'.
    gs_msg-msgv1 = gv_lgnum.
    get_bapiret2 'DE'.
    APPEND es_bapiret2 TO et_logbuch.
  ELSE.
    ev_subrc = 4.
    gs_msg-msgty = 'E'.
    gs_msg-msgno = '060'.
    get_bapiret2 'DE'.
    APPEND es_bapiret2 TO et_logbuch.
    EXIT.
  ENDIF.

* Kombination Lagertyp/Lagerplatz im System vorhanden?
  SELECT SINGLE * FROM lagp
    WHERE lgnum = gv_lgnum
    AND   lgtyp = iv_lgtyp
    AND   lgpla = iv_lgpla.

* Logbucheintrag
  IF sy-subrc EQ 0.
    gs_msg-msgty = 'I'.
    gs_msg-msgno = '073'.
    gs_msg-msgv1 = iv_lgpla.
    gs_msg-msgv2 = iv_lgtyp.
    get_bapiret2 'DE'.
    APPEND es_bapiret2 TO et_logbuch.
  ELSE.
    ev_subrc = 4.
    gs_msg-msgty = 'E'.
    gs_msg-msgno = '074'.
    gs_msg-msgv1 = iv_lgpla.
    gs_msg-msgv2 = iv_lgtyp.
    get_bapiret2 'DE'.
    APPEND es_bapiret2 TO et_logbuch.
    EXIT.
  ENDIF.


************************************************************************
* I N V E N T U R B E L E G S P R Ü F U N G
*
* Ausgangsbasis:  Kein Beleg vorhanden
*                 Beleg vorhanden aber nicht aktiv
*                 Beleg vorhanden und aktiv
*                 Beleg vorhanden - Platz gezählt
*
* Ergebnis: Aktiver Inventurbeleg
************************************************************************

* Inventurbeleg: Status prüfen
  CLEAR: gv_subrc.
  PERFORM check_inv_status_pla   USING     gv_lgnum
                                           iv_lgtyp
                                           iv_lgpla
                                 CHANGING gv_ivnum
                                          gv_ivpos
                                           gv_subrc.

  IF gv_subrc > 3.  "Kein Inventurbeleg vorhanden!
```

```
*   Logbucheintrag
    gs_msg-msgty = 'I'.
    gs_msg-msgno = '058'.
    get_bapiret2 'DE'.
    APPEND es_bapiret2 TO et_logbuch.

*   Inventurbeleg anlegen und aktivieren
    PERFORM create_inv_pla  USING     gv_lgnum
                                      iv_lgtyp
                                      iv_lgpla
                            CHANGING  gv_ivnum.

*   Logbucheintrag
    IF gv_ivnum IS INITIAL.
      ev_subrc = 4.
      gs_msg-msgty = 'E'.
      gs_msg-msgno = '062'.
      get_bapiret2 'DE'.
      APPEND es_bapiret2 TO et_logbuch.
      EXIT.
    ELSE.
      gs_msg-msgty = 'I'.
      gs_msg-msgno = '061'.
      gs_msg-msgv1 = gv_ivnum.
      get_bapiret2 'DE'.
      APPEND es_bapiret2 TO et_logbuch.
    ENDIF.

  ELSEIF gv_subrc = 0.  "Inventurbeleg vorhanden und aktiv!

*   Logbucheintrag
    gs_msg-msgty = 'I'.
    gs_msg-msgno = '063'.
    gs_msg-msgv1 = gv_ivnum.
    get_bapiret2 'DE'.
    APPEND es_bapiret2 TO et_logbuch.

  ELSEIF gv_subrc = 1.  "Inventurbeleg vorhanden aber inaktiv!

*   Logbucheintrag
    gs_msg-msgty = 'I'.
    gs_msg-msgno = '064'.
    gs_msg-msgv1 = gv_ivnum.
    get_bapiret2 'DE'.
    APPEND es_bapiret2 TO et_logbuch.

*   Inventurbeleg sperren
    CLEAR: gv_subrc.
    PERFORM enqueue_link USING gv_lgnum
                               gv_ivnum
                         CHANGING gv_subrc.

*   Logbucheintrag
    IF gv_subrc <> 0.
      ev_subrc = 4.
      gs_msg-msgty = 'E'.
      gs_msg-msgno = '065'.
      gs_msg-msgv1 = gv_ivnum.
      get_bapiret2 'DE'.
      APPEND es_bapiret2 TO et_logbuch.
      EXIT.
    ENDIF.
```

```
*   Inventurbeleg aktivieren
    CLEAR: gv_subrc.
    PERFORM inventur_aktivieren USING    gv_lgnum
                                         gv_ivnum
                                         iv_lgtyp
                               CHANGING gv_subrc.

*   Logbucheintrag
    IF gv_subrc <> 0.
      ev_subrc = 4.
      gs_msg-msgty = 'E'.
      gs_msg-msgno = '066'.
      gs_msg-msgv1 = gv_ivnum.
      get_bapiret2 'DE'.
      APPEND es_bapiret2 TO et_logbuch.
      EXIT.
    ELSE.
      gs_msg-msgty = 'I'.
      gs_msg-msgno = '067'.
      gs_msg-msgv1 = gv_ivnum.
      get_bapiret2 'DE'.
      APPEND es_bapiret2 TO et_logbuch.
    ENDIF.

  ELSEIF gv_subrc = 2 OR gv_subrc = 3. "Platz bereits angezählt!

*   Logbucheintrag
    gs_msg-msgty = 'I'.
    gs_msg-msgno = '068'.
    gs_msg-msgv1 = gv_ivnum.
    get_bapiret2 'DE'.
    APPEND es_bapiret2 TO et_logbuch.

  ENDIF.


************************************************************************
* N U L L Z Ä H L U N G
*
* Ausgangsbasis: Aktiver Inventurbeleg
* Ergebnis: Nullzählung erfasst
************************************************************************

* Inventurposition zur Aktualisierung holen
  CALL FUNCTION 'Z_SCF_FLEXUS_INV_GET_IVPOS'
       EXPORTING
            iv_lgnum  = gv_lgnum
            iv_ivnum  = gv_ivnum
            iv_lgpla  = iv_lgpla
       IMPORTING
            ev_ivpos  = gv_ivpos
       EXCEPTIONS
            not_found = 1
            OTHERS    = 2.

* Logbucheintrag
  IF sy-subrc <> 0.
    ev_subrc = 4.
    gs_msg-msgty = 'E'.
    gs_msg-msgno = '070'.
    get_bapiret2 'DE'.
    APPEND es_bapiret2 TO et_logbuch.
```

```
    EXIT.
  ELSE.
    gs_msg-msgty = 'I'.
    gs_msg-msgno = '069'.
    gs_msg-msgv1 = gv_ivpos.
    get_bapiret2 'DE'.
    APPEND es_bapiret2 TO et_logbuch.
  ENDIF.

* Prüfen ob offene Quants vorhanden sind
  SELECT * FROM linv INTO TABLE lt_linv
           WHERE lgnum = gv_lgnum
           AND   ivnum = gv_ivnum
           AND   ivpos = gv_ivpos
           AND   nanum = '00'
           AND   ( istat = 'N'
           OR      istat = 'A' )
           AND   lgtyp = iv_lgtyp
           AND   lgpla = iv_lgpla.

  CLEAR: gv_lines.
  DESCRIBE TABLE lt_linv LINES gv_lines.
  IF gv_lines NE 0.

*   Nullmengen buchen
    LOOP AT lt_linv INTO ls_linv.

*     Beleg sperren
      CLEAR: gv_subrc.
      PERFORM enqueue_link USING    gv_lgnum
                                    ls_linv-ivnum
                           CHANGING gv_subrc.

*     Logbucheintrag
      IF gv_subrc <> 0.
        ev_subrc = 4.
        gs_msg-msgty = 'E'.
        gs_msg-msgno = '065'.
        gs_msg-msgv1 = gv_ivnum.
        get_bapiret2 'DE'.
        APPEND es_bapiret2 TO et_logbuch.
        EXIT.
      ENDIF.

*     Zählung durchführen
      CLEAR: ls_linv-menge.
      CALL FUNCTION 'Z_SCF_FLEXUS_INV_ZAEHLUNG_WM'
           EXPORTING
                iv_lgnum   = gv_lgnum
                iv_ivnum   = ls_linv-ivnum
                iv_ivpos   = ls_linv-ivpos
                iv_lqnum   = ls_linv-lqnum
                iv_menge   = ls_linv-menge
                iv_meins   = ls_linv-meins
           EXCEPTIONS
                not_found  = 1
                book_error = 2
                OTHERS     = 3.

*     Logbucheintrag
      IF sy-subrc <> 0.
        ev_subrc = 4.
        gs_msg-msgty = 'E'.
```

```
        gs_msg-msgno = '072'.
        get_bapiret2 'DE'.
        APPEND es_bapiret2 TO et_logbuch.
        EXIT.
      ELSE.
        ev_subrc = 0.
        ev_ivnum = ls_linv-ivnum.
        gs_msg-msgty = 'S'.
        gs_msg-msgno = '057'.
        gs_msg-msgv1 = ls_linv-ivnum.
        get_bapiret2 'DE'.
      ENDIF.

    ENDLOOP.

  ELSE.
*   Logbucheintrag
    ev_subrc = 4.
    gs_msg-msgty = 'E'.
    gs_msg-msgno = '071'.
    get_bapiret2 'DE'.
    APPEND es_bapiret2 TO et_logbuch.
    EXIT.
  ENDIF.

ENDFUNCTION.
```

4 Aufgabenstellung und Zielerreichung

4.1 Motivation der Aufgabenstellung durch den Auftraggeber

Die Flexus AG als vergebendes Unternehmen ist Spezialist im Bereich der Online-Anbindung von *RF* (*Radio Frequency*)-*Scannern* an das *WM* (*Warehouse Management)-Modul* der betriebswirtschaftlichen Standardsoftware der Firma SAP. Bei der Integration werden, je nach Kundenanforderung, verschiedene Entwicklungsumgebungen (z.B. SAP Web Application Server, ABAP Workbench, etc.) Software-Komponenten (z.B. SAP Console, SAP Webconsole, etc.) und Entwicklungssprachen (HTML, BHTML, Java, JavaScript, ABAP, etc.) verwendet.

Übergeordnetes Ziel des Unternehmens ist es, vorgefertigte Templates gängiger Transaktionen (z.B. Wareneingang, Inventur, Kommissionieren…), unter Berücksichtigung der erworbenen Erfahrungswerte in vergangenen Projekten, zu erstellen, um weitere Anforderungen der Kunden einfach und rasch umsetzen zu können. Aus Gründen der Flexibilität und Unabhängigkeit soll die vorrangige Entwicklungsumgebung der SAP WAS (Web Application Server) sein. Weiters sollte geprüft werden inwieweit eine Sprachsteuerung einzelner Transaktionen mittels Integration von VoiceXML am Web Application Server umgesetzt werden kann.

Es wurde bereits eine Transaktion zur Kommissionierung mit Nutzung von *Pick by Voice* bei einem Kunden produktiv gesetzt. Diese Transaktion verwendet allerdings die SAP Webconsole. Das heißt, die Ablauflogik wurde mittels ABAP unter Verwendung der ABAP Development Workbench des R/3-Backend-Systems programmiert. Die SAP WEBConsole wandelt den SAP GUI Datenstrom in eine HTML Seite um. Diese HTML Seite wird auf dem Client (in diesem Fall ein PDA) in einem Browser dargestellt. Im Falle von Pick by Voice handelt es sich um einen speziellen Browser (bzw. Client), der die, von der SAP WEBConsole erzeugten, VoiceXML Befehle in Sprachausgaben umwandelt. Als Client wird zurzeit die Sprach-Engine und der XVBrowser der Firma TOP System verwendet.

Der Nachteil durch die Verwendung der SAP Webconsole ist die fehlende Möglichkeit individueller Eingriffe im Bereich Steuerung und Layout, da die HTML-Seiten für den Client vollautomatisch erzeugt werden. Diese Automatik liefert allerdings oft unzureichende Ergebnisse.

4.2 Beschreibung der erreichten Ziele

Die Einzelleistung des IT Praktikums bestand in der selbständigen Entwicklung der gewünschten Beispieltransaktionen. Den aufwendigsten Punkt stellte die Analyse bzw. der selbständige Aufbau des nötigen Know-hows im Bereich X+V (VoiceXML) sowie JavaScript dar, da von Seiten des vergebenden Unternehmens noch wenige Erfahrungswerte vorlagen. Das erworbene Know-how wird in Form der vorliegenden schriftlichen Ausarbeitung (vor allem durch die Beschreibung eines allgemeinen Templates zur Entwicklung von X+V Seiten, siehe Subkapitel 4.3) an das Entwickler-Team der Flexus AG weitergegeben.

Am Beginn des Projekts wurden die einzelnen Möglichkeiten zur Anbindung von Handhelds an SAP mit einander verglichen. Unter Berücksichtigung der IST Situation der Kunden aus systemtechnischer Sicht wurde festgestellt, dass die Anbindung über den Web Application Server der Lösung mit der SAP WEBConsole vorzuziehen ist, da die meisten Kunden der Flexus AG den WAS bereits im Einsatz haben oder zu mindest Bestrebungen vorliegen, diese Entwicklungsumgebung kurzfristig produktiv zu setzen. Für die Entwickler bedeutet die Nutzung des WAS weitaus mehr Spielraum in funktioneller und visueller Hinsicht.

Im nächsten Schritt wurde der X+V Standard analysiert, um festzustellen, welche Mindestanforderungen an die zu entwickelten Seiten gestellt werden. Außerdem wurde eruiert, welche Software und Tools zurzeit auf dem Markt verfügbar sind. Um die Möglichkeiten des X+V Standards auch tatsächlich nutzen zu können musste im Bereich VoiceXML und JavaScript das nötige Know-how aufgebaut werden. Zuletzt wurde festgestellt, dass die Entwicklung der X+V Seiten auf dem WAS zu keinerlei Problemen führt und analog zur automatischen Generierung durch die SAP WEBConsole durchgeführt werden kann.

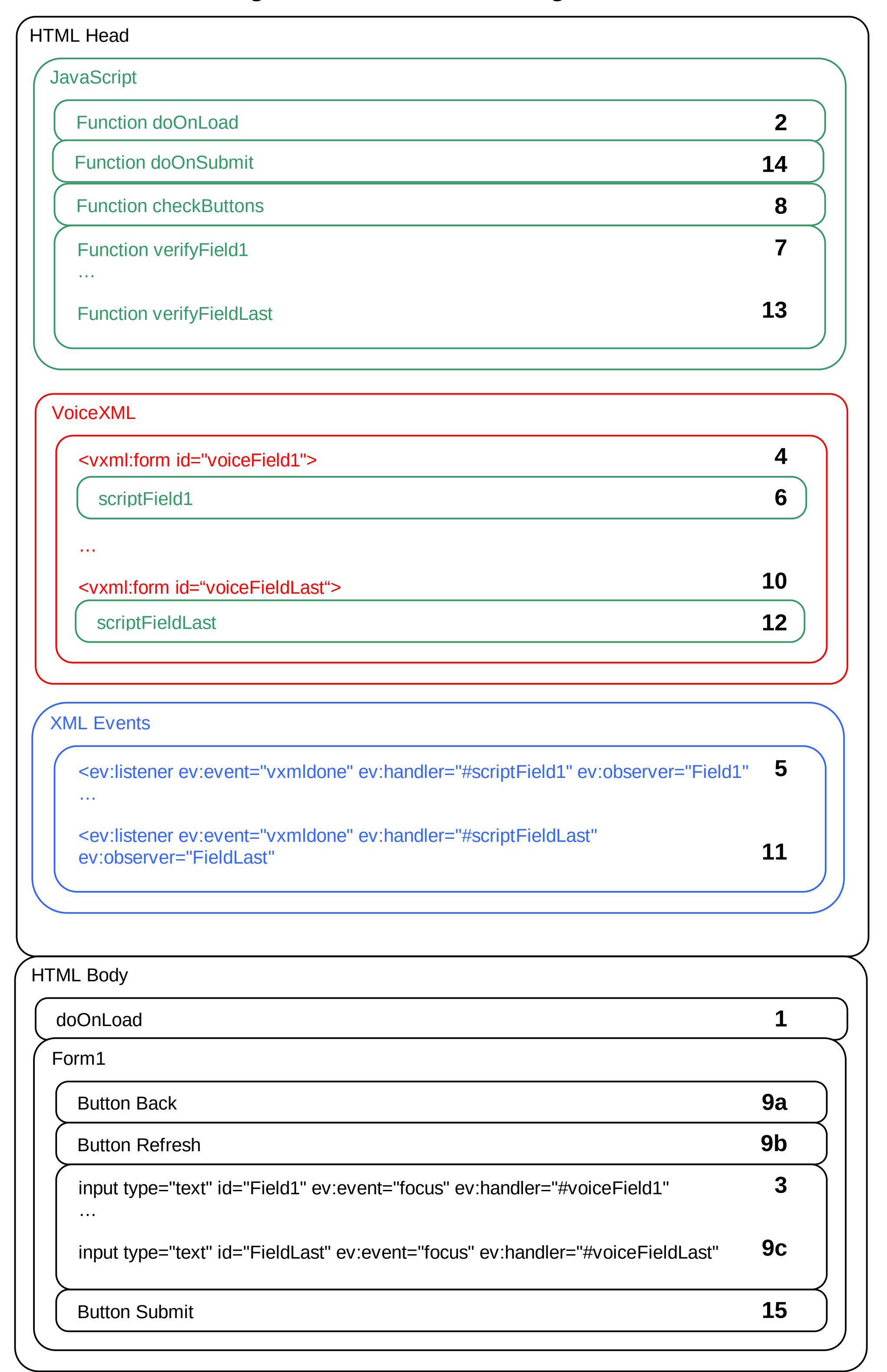
Weiters wurde geprüft inwieweit der XVBrowser der Firma TOP System den X+V Standard unterstützt und welche Unterschiede zu nichtkommerziellen multimodalen Browsern wie Netfront oder Opera vorliegen. Der XVBrowser stellt zwar keinen vollständigen multimodalen Browser dar, da nicht alle VoiceXML Tags unterstützt werden, allerdings können, sofern die Einschränkungen bekannt sind, mit hoher Wahrscheinlichkeit alle Programmanforderungen umgesetzt werden. Vor allem im Bereich JavaScript bietet der XVBrowser nur eingeschränkte Funktionalität. Außerdem ist anzumerken, dass, im Gegensatz zu Netfront oder Opera, ein umfangreiches Sprachtraining durchzuführen ist, welches einer zusätzlichen Schulung der zukünftigen Anwender bedarf.

Zum Abschluss des IT Praktikums wurde darauf Wert gelegt, die erarbeiteten Informationen für die Entwickler der Flexus AG bereitzustellen und einen Nutzen in der praktischen Entwicklungstätigkeit abzuleiten. Dies wurde in Form der Erstellung eines Templates als Vorschlag zur raschen und komfortablen Entwicklung von X+V Seiten in zukünftigen Projekten durchgeführt. Dieses Template soll Teil des firmeninternen Styleguides für SAP Kundenentwicklungen der Flexus AG sein. Durch den einheitlichen Aufbau aller X+V Seiten soll eine kostengünstigere Entwicklung und im Anschluss daran eine einfachere Wartung gewährleistet werden.

4.3 Template zur Entwicklung von sprachgesteuerten Business Server Pages

Zum besseren Verständnis wurde eine schematische Darstellung ausgearbeitet die den grundsätzlichen Aufbau einer X+V Seite beschreiben soll (vgl. Abbildung 31). Zur besseren Übersicht werden die verschiedenen Sprachteile der X+V Seite mit unterschiedlichen Farben dargestellt. Die beigefügten Ziffern am rechten Rand geben die vorgesehene Reihenfolge der Verarbeitung der einzelnen Blöcke an.

Abbildung 31: Schematische Darstellung einer X+V Seite



Quelle: Eigene Abbildung

Aus der vorangegangenen schematischen Darstellung ergibt sich der folgende Quelltext einer einfachen X+V Seite (mit zwei Eingabefeldern) als Ausgangsbasis zur Entwicklung von sprachgesteuerten Business Server Pages.

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:vxml="http://www.w3.org/2001/vxml"
      xmlns:ev="http://www.w3.org/2001/xml-events">
<head>
      <title>Template</title>


<script type="text/javascript">
      var recognized="" ;


      function doOnLoad() {
            document.form1.Field1.focus() ;
      }

      function doOnSubmit() {
            document.getElementById('Book').click();
      }

      function checkButtons(ok_code){
            if (ok_code=="back"){
                  document.form1.Back.click();
                  return false;
            }else if (ok_code=="refresh"){
                  document.form1.Refr.click();
                  return false;
            }else
                  return true;
      }

      function verifyField1(recognized) {
            if (checkButtons(recognized))
                  return true;
            else
                  return false;
      }

      function verifyFieldLast(recognized) {
            return true;
      }
</script>


<vxml:form id="voiceField1">
      <vxml:field name="v_Field1">
            <vxml:prompt>
                  This ist text for Field 1
            </vxml:prompt>
            <vxml:grammar>
                  <![CDATA[
                        #JSGF V1.0;
                        grammar gram1;
                        public <gram1> = test | back | refresh;
                  ]]>
            </vxml:grammar>
            <vxml:catch event=" help">
                  This is text to help the user.
```

```
          </vxml:catch>
          <vxml:filled>
               <vxml:assign name="recognized" expr="v_Field1"/>
               confirm <vxml:value expr="v_Field1"/>
          </vxml:filled>
     </vxml:field>
</vxml:form>

<script type="text/javascript" id="scriptField1">
     if(recognized!="") {
          if(verifyField1(recognized)) {
               document.form1.Field1.value=recognized ;
               document.form1.FieldLast.focus() ;
          }
     }
</script>


<vxml:form id="voiceFieldLast">
     <vxml:field name="v_FieldLast">
          <vxml:prompt>
               This ist text for the last Field
          </vxml:prompt>
          <vxml:grammar>
               <![CDATA[
                    #JSGF V1.0;
                    grammar gramLast;
                    public <gramLast> = next;
               ]]>
          </vxml:grammar>
          <vxml:catch event="help">
               The user should say "NEXT".
          </vxml:catch>
          <vxml:filled>
               <vxml:assign name="recognized" expr="v_FieldLast"/>
          </vxml:filled>
     </vxml:field>
</vxml:form>

<script type="text/javascript" id="scriptFieldLast">
     if(recognized!="") {
          if(verifyFieldLast(recognized)) {
               if(recognized=="next")
                    doOnSubmit() ;
          }
     }
</script>

<ev:listener ev:event="vxmldone" ev:handler="#scriptField1"
     ev:observer="Field1" ev:propagate="stop" />
<ev:listener ev:event="vxmldone" ev:handler="#scriptFieldLast"
     ev:observer="FieldLast" ev:propagate="stop" />

</head>


<body onload="javascript:doOnLoad();">

<form name="form1" method="post">

     <table cellpadding="5" align="center">
          <tr bgcolor="#BBBBFF">
               <td align="center">
```

```
                    <input style="text-align:center; width:80" type="button"
                    name="Back" value="Back"
                    onClick="self.location.href='template.html'"/></td>
               <td align="center">
                    <input style="text-align:center; width:80" type="button"
                    name="Refr" value="Refresh"
                    onClick="self.location.href='template.html'"/></td> </tr>
          <tr bgcolor="#BBBBFF">
               <td>Field 1:</td>
               <td>
                    <input type="text" name="Field1" id="Field1"
                    ev:event="focus" ev:handler="#voiceField1"/> (‘test’)
               </td>
          </tr>
          <tr bgcolor="#BBBBFF">
               <td>Last Field:</td>
               <td>
                    <input type="text" name="FieldLast" id="FieldLast"
                    ev:event="focus" ev:handler="#voiceFieldLast"/> (‘next’)
               </td>
          </tr>
          <tr bgcolor="#BBBBFF">
               <td colspan="2" align="center">
                    <input style="text-align:center; width:150" type="submit"
                    name="onInputProcessing(BOOK)" id="Book" value="Submit"/>
               </td>
          </tr>
     </table>

</form>

</body>

</html>
```

Literatur- und Quellenverzeichnis

ACCESS (2005): NetFront™ and Voice Recognition. Verfügbar von: http://www.access-us-inc.com/Products/client-side/Prod_NetFront_nf_xhtml.html, Abfragedatum: 13. September 2005.

Bruck Technologies (o.J.): Handhelds mit Funk-LAN-Koppelung unterstützen den Außendienst. Verfügbar von: http://www.brucktech.com/index.php?cmenue_dokumentenid=107, Abfragedatum: 9. November 2005.

Flexus AG (2004): SAP Kommunikationstechniken. Präsentationsunterlagen, Würzburg.

Giordano, M. und Hummel J. (Hrsg.) (2005): *Mobile Business, Vom Geschäftsmodell zum Geschäftserfolg*. Wiesbaden: Gabler Verlag.

IBM (o.J.): Multimodal, Why IBM? – Leadership in multimodal. Verfügbar von: http://www-306.ibm.com/software/pervasive/multimodal/, Abfragedatum: 13. September 2005.

IBM (2003a): Multimodal Application Design Issues. White Paper, o.O.

IBM (2003b): Developing Multimodal Applications using XHTML+Voice. White Paper, New York.

IBM (2004): XHTML+Voice Programmer's Guide. White Paper, o.O.

Kafka, G. (2005): *WLAN Technik, Standards, Planung und Sicherheit für Wireless LAN*. München: Carl Hanser Verlag.

Marlovits, G. H. (2002): PDAs als mobile Datenerfassungsgeräte am Beispiel der Implementierung eines Bibliotheksverwaltungssystems. Diplomarbeit an der Wirtschaftsuniversität Wien. Wien

Opera (2005): Let There Be Sound Too: Adding Voice to XHTML. Verfügbar von: http://my.opera.com/community/dev/voice/, Abfragedatum: 13. September 2005.

Roth, J. (2002): *Mobile Computing*. Heidelberg: dpunkt-Verlag.

SAP AG (2002a): Atomic Austria realisiert mobile Funk-Scanner-Lösung. Verfügbar von: http://www11.sap.com/germany/media/Atomic.pdf, Abfragedatum: 10. November 2005.

SAP AG (2002b): Lagerverwaltung mit mySAP™ Supply Chain Management. Verfügbar von: http://www70.sap.com/germany/media/50060109.pdf, Abfragedatum: 12. September 2005.

SAP AG (2004): *Web-Enabled SAPConsole and Voice Recognition, Cookbook*. White Paper, Walldorf.

SAP Online Documentation (2005a): Web Application Server (BC). Verfügbar von: http://help.sap.com/saphelp_erp2004/helpdata/de/35/2cd77bd7705394e10000009b387c12/frameset.htm, Abfragedatum: 12. September 2005.

Serloth, A. (2005): Wireless Security: Sicherer Funken. *Peter F. Mayer, Magazin für Infrastruktur und Technologie*, Nr. 7 November 2005, S.26.

Steimer, F. L., Maier, I. und Spinner, M. (2001): *mCommerce, Einsatz und Anwendung von portablen Geräten für mobilen eCommerce*. München: Addison-Wesley Verlag.

Topsystem (2005): *topSPEECH Lydia®-PDA. Specific extensions for the SAP WebConsole*. Unveröffentlichtes Manuskript, Würselen.

W3C (2003): XML Events, An Events Syntax for XML. Verfügbar von: http://www.w3.org/TR/xml-events/, Abfragedatum: 13. September 2005.

W3C (2004a): XHTML + Voice Profile 1.2. Verfügbar von: http://www.voicexml.org/specs/multimodal/x+v/12/, Abfragedatum: 13. September 2005.

W3C (2004b): Voice Extensible Markup Language (VoiceXML) Version 2.0. Verfügbar von: http://www.w3.org/TR/voicexml20/, Abfragedatum: 13. September 2005.

W3C (2005): XHTML™ 2.0, W3C Working Draft 27 May 2005. Verfügbar von: http://www.w3.org/TR/2005/WD-xhtml2-20050527/, Abfragedatum: 13. September 2005.

Wikipedia (2005b): Sprachsynthese. Verfügbar von: http://de.wikipedia.org/wiki/Text-to-Speech, Abfragedatum: 12. September 2005.

Wikipedia (2005c): Spracherkennung. Verfügbar von: http://de.wikipedia.org/wiki/Spracherkennung, Abfragedatum: 12. September 2005.

Abbildungs- und Tabellenverzeichnis